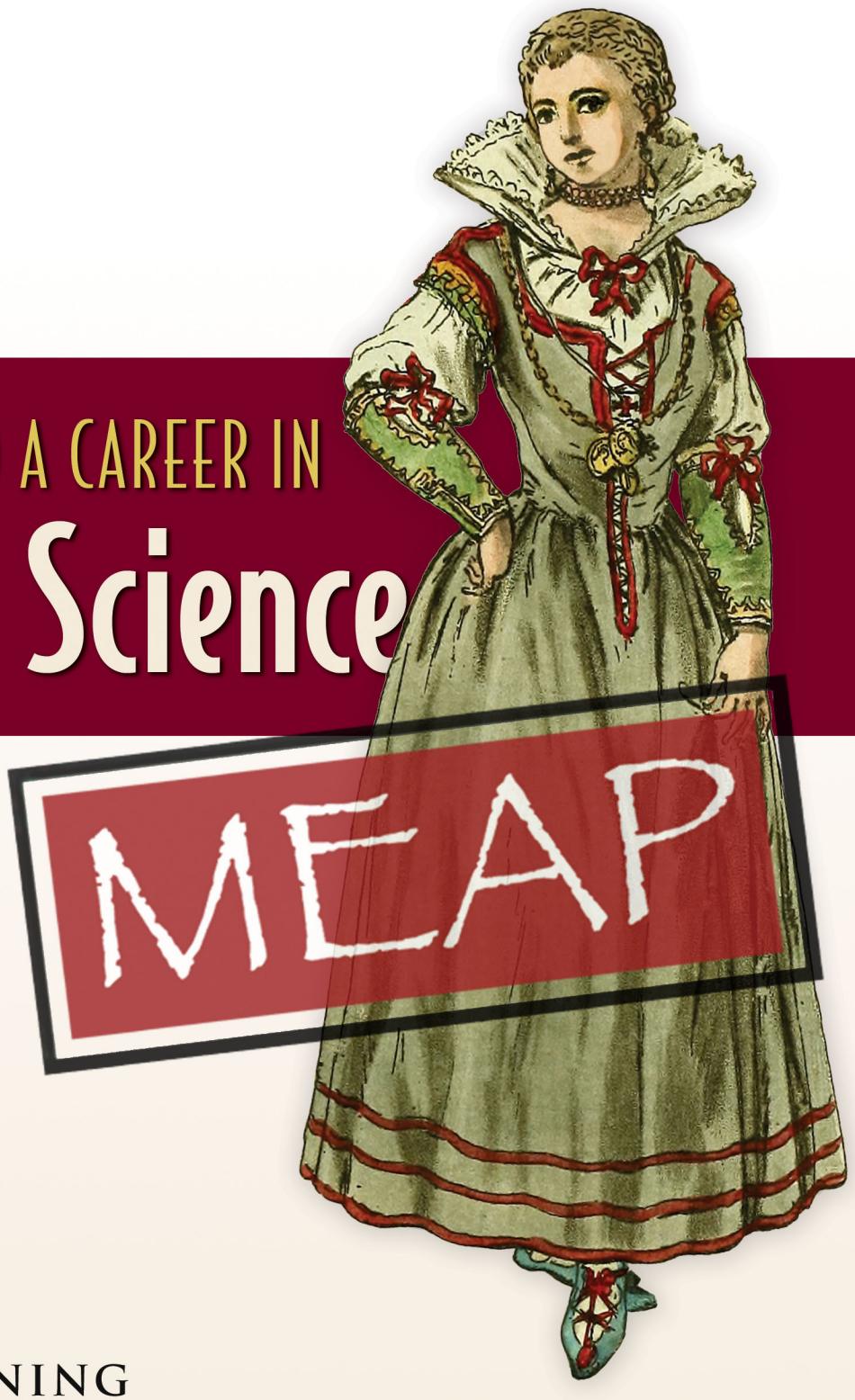
BUILD A CAREER IN Data Science



Emily Robinson
Jacqueline Nolis

MANNING

MEAP Edition
Manning Early Access Program
Build a Career in Data Science
Version 3

For more information on this and other Manning titles go to
manning.com

welcome

Thank you for purchasing the MEAP for *Building a Career in Data Science!*

Data science is a fast-growing and exciting field, with many companies building out their teams and more and more people wanting to join it. But how do you go from making the decision to become a data scientist to actually becoming one? And not only that, but someone who excels in their role?

The idea for this book started when we realized how many questions we were getting from people on their data science career path. We each knew of some good blog posts that have been written on various topics, but we couldn't suggest a book to read that delivered a full how to guide on how to start and grow a data science career. Well, until now.

This book is written for people who either want to learn how to start a career in data science or grow their existing one. Part 1 of the book covers how to get into the field of data science—what it is, how it varies by company, and how to build your skills and create a portfolio of projects. Part 2 covers getting a job, starting from how to find promising positions to writing a stellar resume and cover letter, interviewing well, and negotiating an offer. Part 3 is about what to expect and how to perform well once you're in the job, including making analyses, deploying models into production, and dealing with stakeholders. Finally, part 4 will help you grow in your role, advising you on how to handle failed projects, contribute to open-source, and choosing between management and individual contributor roles.

The book is opinionated—rather than it being a collection of provable facts, we've focused on including things we've personally noticed can really help or harm a career (including our own). At every turn we've strived to be actionable—the goal of the book has been that the reader will walk away with new tools and ideas to try. At the end of each chapter we've included an interview with a prominent data scientist whose career has highlighted the lessons of the chapter. While at time we discuss technical topics, we've kept the book technologically agnostic to focus on the career components.

As you read the book, we would love to hear what you've found useful or what you've wished we went deeper on. If you have any questions, comments, or suggestions, please share them in Manning's liveBook Discussion Forum for our book.

—Emily Robinson and Jacqueline Nolis

brief contents

A Example interview questions

Part 1: Getting started with data science

If you do a Google search for "how to become a data scientist," you'll likely be confronted with a laundry list of skills, from statistical modeling to programming Python through communicating effectively and making presentations. One job description might describe a role close to a statistician, where another is looking for someone with a master's in computer science. When you look for ways to gain those skills, you'll find options ranging from going back for a master's degree to doing a bootcamp to starting to do data analysis in your current job. Put together, all these different combinations of paths can feel insurmountable, especially to people not yet certain they even want to be a data scientist.

The good news is there is not a single data scientist who knows all these skills. While data scientists share a common foundation of knowledge, each have their own specialty to the point that many couldn't swap jobs. The first part of this book is here to help you understand what all these different types of data scientist are and how to make the best decisions to help your career get started. By the end of this part you should be prepared with the skills and understanding to start your job search for a data science role.

Chapter 1 covers the basics of data science, including the skills you need for the job and the different types of data scientist. Chapter 2 goes into detail about the role of a data scientist at five different types of company to help you better understand what the job will be like. Chapter 3 covers the paths to getting those skills required for being a data scientist and the advantages and disadvantages of each. Finally, Chapter 4 covers how to create a portfolio of data science projects to get hands on experience doing data science and create projects to show potential employers.

1

What is data science?

This chapter covers

- The three main areas of data science: mathematics/statistics, databases/programming, and business understanding
- The different types of data science jobs

"The sexiest job of the 21st century." "The best job in America." Data Scientist, a title that didn't even exist before 2008, is now the position employers can't hire enough of and job seekers strive to become. There's good reason for the hype - data science is a hugely growing field with a median salary of over $100,000 in the United States in 2017. At a good company, data scientists enjoy a lot of autonomy and are constantly learning new things. You use your skills to solve significant problems: working with doctors to analyze drug trials, helping a sports team pick their new draftees, or redesigning the pricing model for a widget business. Finally, as we will discuss in Chapter 3, there's no one way to become a data scientist. People come from all different backgrounds, so you're not limited based on what you chose to study as an undergraduate.

But not all data science jobs are perfect. Both companies and job seekers can have unrealistic expectations. Companies new to data science may think one person can solve all their problems with data. When a data scientist is finally hired, they can be faced with a never-ending to-do list of requests. They might be tasked with immediately implementing a machine learning system when there's been no work to prepare or clean the data. There may be no one to mentor, guide them, or even empathize with the problems they face. We'll discuss these issues more in Chapters 5 and 7, where we'll help you avoid joining companies that are likely to be a bad fit for a new data scientist, and in Chapter 9, where we'll advise you on what to do if you end up in a negative situation.

On the other side, job seekers may think there will never be a dull moment in their new career. They may expect that stakeholders will routinely follow their recommendations, that data engineers can immediately fix any data quality issues, and that they'll get the fastest computing resources available to implement their models. In reality, data scientists spend a lot of time cleaning and preparing data and managing the expectations and priorities of other teams. Projects won't always work out. Senior management may make unrealistic promises to clients about what your data science models can deliver. A person's main job may be to work with an archaic data system that's impossible to automate and requires hours of mind-numbing work each week just to clean up the data. You may notice lots of statistical or technical mistakes in legacy analyses that have real consequences, but no one is interested and you're so overloaded with work you have no time to try to fix them. A data scientist may be asked to prepare reports that support what senior management has already decided, where if you give an independent answer you worry you may be fired.

This book is here to guide you through the process of becoming a data scientist and developing your career. We want to ensure that you, the reader, get all of those great parts of being a data scientist and avoid most of the pitfalls. Maybe you're working in an adjacent field like marketing analytics and are wondering about how to make the switch. Or maybe you're already a data scientist, but you're looking for a new job and don't think you approached your first job search well. Or you want to further your career by speaking at conferences, contributing to open source, or becoming an independent consultant. Whatever your level, we're confident you'll find this book helpful.

In these first four chapters, we'll cover the main opportunities for gaining data science skills and how to build a portfolio to get around the paradox of "needing experience to get experience." In part 2, you'll learn how to write a cover letter and resume that will get you an interview, but also how to build your network to get a referral. We'll cover negotiation strategies that research has shown will get you the best offer possible.

When you're in a data science job, you'll be writing analyses, working with stakeholders, and maybe even putting a model in production. Part 3 will help you understand what all of those processes look like and how to set yourself up for success. In part 4, you'll find strategies to pick yourself back up when a project inevitably fails. And when you're ready, we're here to guide you through the decision of where to take your career - management, staying an individual contributor, or even striking out as an independent consultant.

But before we begin that journey, we need to be clear on what a data scientist is and what is the work they do. Data science is a broad field covering many different types of work, and the better you understand the differences between those areas, the better you can grow in them.

1.1 What is data science?

Data science is the practice of using data to try to understand and solve real-world problems. This isn't exactly new- people have been analyzing sales figures and trends since the invention

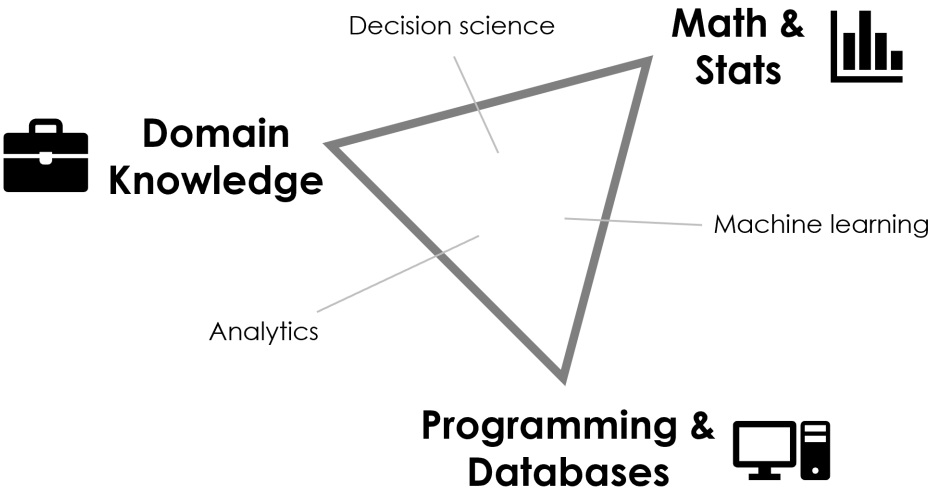
of the zero. In the last decade, however, we have gained access to exponentially more data than existed before. The advent of computers has assisted in the generation of all that data, but computing is also our only way to process the mounds of information. With computer code, a data scientist can transform or aggregate data, run statistical analyses, or train machine learning models. The output of this code may be a report or dashboard for human consumption, or it could be a machine learning model that will be deployed to run continuously.

For example, if a retail company is having trouble deciding where to put a new store, they may call a data scientist in to do an analysis. The data scientist could look at historical data of locations where online orders are shipped to understand where the customer demand is. They may also combine that customer location data with demographic and income information for those localities from census records. With these data sets, they could find the optimal place for the new store and create a PowerPoint presentation to present their recommendation to the company's VP of Retail Operations.

In another situation, that same retail company may want to increase their online order sizes by recommending items to customers while they shop. A data scientist could load the historical web order data and create a machine learning model that, given a set of items currently in the cart, predicts the best item to recommend to the shopper. After creating that model, the data scientist would work with the company's engineering team so that every time a customer is shopping, the new machine learning model will serve up the recommended items.

When many people start looking into data science, one challenge they face is being overwhelmed by the amount of things they need to learn. There's coding (but which language?), statistics (but which methods are most important in practice and which are largely academic?), machine learning (but how is machine learning different than statistics or AI?), and the domain knowledge of whatever industry they want to work in. This is all in addition to business skills like effectively communicating results to audiences ranging from other data scientists to the CEO. This anxiety can be exacerbated by job postings that ask for a PhD, multiple years of data science experience, and expertise in a laundry list of statistical and programming methods. How can you possibly learn all these skills? Which ones should you start with? What are the basics?

If you've looked into the different areas of data science, you may be familiar with Drew Conway's popular data science Venn diagram. In his opinion (at the time of its creation), data science falls in the intersection of math and statistical knowledge, expertise in a domain, and "hacking" skills (i.e. coding). This image is often used as the cornerstone of defining what a data scientist is. From our perspective, the components of data science are slightly different than what he proposed:



We've changed it to a triangle because it's not that you either have a skill or you don't, it's that you may possess it to a different extent than others in the field. While it's true that all three skills are fundamental and you need to have some of each, you don't need to be an expert in all of them. We put within the triangle different types of data science specialties. These don't always map one-to-one with job titles; even when they do, sometimes different companies will call them something different.

So what do each of these components mean?

1.1.1 Mathematics/statistics

At the basic level, mathematics and statistics knowledge is data literacy. We break that down into three different levels of knowledge:

- **That techniques exist**. If you don't know something is possible, you can't use it. For instance, if a data scientist was trying to group similar customers together, knowing that there are statistical methods (called "clustering") to do this would be the first step.
- **How to apply them**. While a data scientist may know about many techniques, they also need to be able to understand the complexities of applying them. That means not only how to write code to apply the method, but also how to configure them. If the data scientist wants to use a method like "*k*-means clustering" to group the customers, they would need to understand how to do a *k*-means clustering in a programming language like R or Python. They would also need to understand how to adjust parameters of the method, like how to choose how many groups to create.
- **How to choose which to try**. Because there are so many possible techniques that

can be used in data science, it's important that the data scientist is able to quickly assess if a technique would work well or not. In our customer grouping example, even once you've focused on clustering, there are dozens of different methods and algorithms. Rather than trying each one, a data scientist needs to be able to rule out methods quickly and focus on just a few.

These sorts of skills are being constantly used within a data science role. To take a different example, let's say you work at an e-commerce company. Your business partner might be interested in what countries have the highest average order value. If you have the data available, this is an easy question to answer. But rather than simply presenting this information and letting them draw their own conclusions, you could dig deeper. If you only have one order from country A that was $100, and a thousand orders from country B with an average of $75, it is correct that country A has the higher average order value. But would you be confident in saying that means they should definitely invest in advertising in country A to increase the number of orders? Probably not - you only have one data point, and maybe it's an outlier. If country A had 500 orders instead, you might use a statistical test to see if the order value was significantly different, meaning that if there really was no difference between A and B on this measure, it would be unlikely you'd have seen the difference you did. In this one paragraph-long example, many different assessments were made on what approaches were sensible, what should be considered, and what results were deemed unimportant.

1.1.2 Databases/programming

"Programming and databases" refers to the ability to pull data from company databases and to write clean, efficient, maintainable code. These skills are in many ways similar to what a software developer has to know, except data scientists have to write code that does open-ended analysis rather than producing a predefined output. Each company's data stack is unique, so there is no one set of technical skills that are required for a data scientist. But broadly, you'll need to know how to get data from a database and how to clean, manipulate, summarize, visualize, and share data.

For most data science jobs, R or Python will be the main language you use. R is a programming language that has its roots in statistics, so it's generally strongest in statistical analysis and modeling, visualization, and generating reports with results. Python is a programming language that started as a general software development language and has become extremely popular in data science. Python is known for being better at working with large data sets, doing machine learning, and powering real-time algorithms (like Amazon's recommendation engines). But thanks to the work of many contributors, the two languages capabilities are now at near parity. Data scientists are successfully using R to make machine learning models that are run millions of times a week, and they are also making clean, presentable statistical analyses in Python.

R and Python are the most popular languages for data science for a couple of reasons. They're open-source, meaning they're free and many people, not just one company or one

group, contribute code you can use. They have many packages or libraries (sets of code) for doing data collection, manipulation and visualization, statistical analysis, and machine learning. And importantly, because they each have such a large following, it's easy for data scientists to find help when they run into issues. While some companies still use SAS, SPSS, STATA, MATLAB, or other paid programs, many of them are actually starting to move to use R or Python instead.

While most data science analysis is done in R or Python, you'll often need to work with a database to get the data. This is where the language SQL comes in. SQL is the programming language that most databases use to manipulate data within them or to extract it. For example, consider a data scientist who wanted to analyze the hundreds of millions of records of customer orders in a company to forecast how the orders per day will change over time. Here they would likely first write a SQL query to get the number of orders each day. Then they would take those daily order counts and run a statistical forecast in R or Python. For this reason, SQL is extremely popular in the data science community, and it's difficult to get too far without knowing it.

Finally, another core skill is using version control. Version control is a method of keeping track how code is changing over time. Version control lets you store your files, revert them back to a previous time, and see who changed what file, how, and when. It's extremely important for data science and software engineering because if someone accidentally changes a file that breaks your code, you want the ability to revert or see what changed.

Git is by far the most commonly used system for version control, often used in conjunction with GitHub, a web-based hosting service for git. Git allows you to save ("commit") your changes as well as go back and see the whole history of the project and how it has changed with each commit. If two people are working on the same file separately, git makes sure that no one's work is ever accidentally deleted or overwritten. At many companies, especially those with strong engineering teams, you'll need to use git if you want to share your code or put something in production.

Can you be a data scientist without programming?

It's possible to do a lot of data work using only Excel, Tableau, or other business intelligence tools that have graphical interfaces. While you're not writing code, these tools claim to have much of the same functionality as languages like R or Python, and many data scientist do use them sometimes. But can this be a complete data science toolkit? We say no. Practically, very few companies have a data science team where you wouldn't need to program. But even if that wasn't the case, programming has advantages over these tools.

The first advantage of programming is reproducibility. When you write code instead of using point-and-click software, you're able to rerun it whenever your data changes, whether that's every day or in six months. This also ties into version control - instead of re-naming your file every time your code changes so you can always go back to an old piece, you're able to keep one file but see the entire history.

The second is flexibility. If Tableau doesn't have a type of graph available, you won't be able to create it. With programming, you can write your own code to make something the creators and maintainers of a tool never thought of.

The third and final advantage for open-source languages like Python and R are the community contributions. Thousands of people create packages, or bundles of code, and publish it openly on GitHub. That means you can download their code and use it for your own problems. You're not reliant on one company or group of people to add features.

1.1.3 Business understanding

"Any sufficiently advanced technology is indistinguishable from magic." -Arthur C. Clarke.

Businesses have, to put it mildly, a varying understanding of how data science works. Often, management would just like something done and turn to their data science unicorns to make that thing happen. A core skill in data science is knowing how to translate a business situation into a data question, find the data answer, and finally deliver back the business answer. For example, a business person might ask "why are our customers leaving?" But it's not like there is a "why-are-customers-leaving" Python package you can import—it's up to the data scientist to deduce how to answer that question with data.

Business understanding is where your data science ideals meet head-on with the practicalities of the real world. It's not enough to want a specific piece of information without knowing how the data is stored and updated at your specific company. If your company is a subscription service, where does the data live? If someone changes their subscription, what happens? Does their row get updated or is another row added to the table? Are there errors or inconsistencies in the data you need to work around? If you don't know the answers to these questions, you won't be able to give an accurate answer to the basic question of, "How many subscribers did we have on March 2, 2019?"

Business understanding also helps you to know what questions to ask. Hearing from an stakeholder, "What should we do next," is a little like being asked, "Why do we not have more money?" It's a question that begs more questions. Developing an understanding of the core business (as well as the personalities involved) can help you parse the situation better. Maybe follow-up with "Which product line are you looking for guidance regarding?" or "Would you like to see more participation from a certain sector of our audience?"

Part of business understanding is also developing general business skills like being able to tailor your presentations and reports to different audiences. Sometimes you will be discussing a better methodology with a room full of statistics PhDs and sometimes you will be in front of a vice president who hasn't taken a math class in twenty years. You need to inform your audience without either talking down or overcomplicating.

Finally, as you become more senior, part of your job is to identify where the business could benefit from data science and/or machine learning. If you've wanted to build a prediction system for your company, but have never had management support, becoming part of the management team can help solve that. A senior data scientist will be on the lookout for places they can implement machine learning, as they know its limitations and capabilities and which kind of tasks would benefit from automation.

Will data science disappear?

Underlying the question of whether data science will still be around in a decade or two are two main concerns - that the job will become automated and that data science is overhyped and the job market bubble will pop.

It's true that certain parts of the data science pipeline can be automated. Automated Machine Learning, or AutoML, can compare the performance of different models and perform certain parts of data preparation (like scaling variables). But these are just a small part of the data science process. For example, you'll often need to create the data yourself - it's very rare that there is perfectly clean data waiting for you. Creating the data will usually involve talking with other people, such as user-experience researchers or engineers, who will conduct the survey or log the user actions that can drive your analysis.

Regarding the possibility of a pop in the job market bubble, a good comparison is software engineering in the 1980s. As computers grew cheaper, faster, and more common, there were concerns that soon a computer could do everything and there would be no need for programmers. But the opposite happened and there are now more than 1.2 million software engineers in the United States (Bureau of Labor Statistics, U.S. Department of Labor, Occupational Outlook Handbook, Software Developers, on the Internet at https://www.bls.gov/ooh/computer-and-information-technology/software-developers.htm). While things like the title "webmaster" did disappear, there are now more people working on website development, maintenance, and improvement than ever.

We believe that there will be more specialization within data science, which might lead to the general title of data scientist disappearing. But many companies are just in the early stages of learning how they can leverage data science.

1.2 Different types of data science jobs

You can mix and match those three core skills of data science into a number of jobs, all which have some justification to be called a data scientist. In our perspective, there are three main ways that these skills get mixed together: analytics, machine learning, and decision science. Each of those three areas serves a different purpose to the company and fundamentally delivers a different thing.

When looking for data science jobs, you should pay less attention to the job titles and more to the job descriptions and what they ask you in the interviews. Look at the background of people in those roles – what were their previous jobs and degrees? You may find that people working similar sounding jobs have totally different titles, or that people with the same data scientist title do totally different things. As we talk about the different types of data science jobs, remember that the actual titles used at companies may vary.

1.2.1 Analytics

An analyst takes data and puts it in front of the right people. For example, after a company sets its yearly goals, you might put that in a dashboard so they can track the progress every week. You could build in features so that they could easily break down the numbers by country or product type. This work involves a lot of data cleaning and preparation, but generally less work to interpret the data. While you should be able to spot and fix data quality issues, the primary person making decisions with this data is the business partner. Thus the job of an analyst is to take data from within the company, format it and arrange it effectively, and deliver that data to others.

Because there's not a lot of statistics and machine learning in this role, some people and companies would consider this role outside of the field of data science. But there is a lot of work like devising meaningful visualizations and deciding on particular data transformations that are the same skills as required in the other types of data science roles. For example, an analyst may be given a task like "create an automated dashboard that shows how our number of subscribers is changing over time and lets us filter the data to just subscribers of specific products or in specific geographical regions." The analyst would have to find the appropriate data within the company, figure out how to transform the data appropriately (like change it from daily to weekly new subscriptions), then create a meaningful set of dashboards that are visually compelling and automatically update each day without errors.

Short rule: an analyst creates *dashboards and reports that deliver data.*

1.2.2 Machine Learning

A machine learning engineer develops machine learning models and puts them in production, where they will run continuously. They may optimize the ranking algorithm for the search results of an e-commerce site, create a recommendation system, or monitor a model in production to make sure its performance hasn't degraded since it was deployed. A machine learning engineer spends less time on things like creating visualizations that will convince people of something, and more time doing the programming work of data science.

A big difference between this role and other types of data science positions is that your work output is primarily for machine consumption—you create machine learning models that get turned into APIs, or application programming interfaces, for other machines. In many ways you will be closer to a software developer than other data science roles. While it's good for any data scientist to follow best coding practices, as a machine learning engineer you must. Your code must be performant, tested, and written so that other people will be able to work with it. For this reason, many machine learning engineers come from a computer science background.

In a machine learning engineer role, a person may be asked to create a machine learning model that can in real time predict the probability that a customer on the website will actually finish their order. The machine learning engineer would have to find historical data in the company, train a machine learning model on it, take that model and turn it into an API, then deploy the API so the website can run the model. If that model stops working for some reason, the machine learning engineer will be called to fix it.

Short rule: a machine learning engineer creates *models that get run continuously.*

1.2.3 Decision Science

A decision scientist turns a company's raw data into information that helps the company to make decisions. This work relies on having a deep understanding of different mathematical and statistical methods and a familiarity with business decision-making. Furthermore, decision scientists have to be able to make compelling visualizations and tables so that the non-

technical people they talk to will understand their analysis. While a decision scientist does plenty of programming, their work generally only gets run once to make a particular analysis. That means they can get away with having code that's inefficient or difficult to maintain.

A decision scientist must understand the needs of the other people within the company and figure out how to generate constructive information. For example, a marketing director may ask a decision scientist to help decide which types of products they should highlight in their holiday gift guide. A decision scientist might investigate what products have sold well without having been featured in the gift guide, talk to the user research team about conducting a survey, and use principles of behavioral science to do an analysis to come up with the optimal items to suggest. The final result is likely to be a PowerPoint presentation or report to be shared with product managers, vice presidents, and other business people.

A decision scientist often uses their knowledge of statistics to help the company make decisions under uncertainty. For example, a decision scientist could be responsible for running their company's experimentation analytics system. Many companies run online experiments, or A/B tests, to measure whether a change is effective. This change could be as simple as adding a new button or as complicated as changing the ranking system of search results or completely redesigning a page. During an A/B test, visitors are randomly assigned to one of two or more conditions (e.g. half to the old version of the homepage, the "control," and half to the new, "the treatment"). Their actions after they enter the experiment are then compared to see if those in the treatment have a higher rate of doing desirable actions like buying products.

Because of randomness, it's rare that the metrics in the control and treatment are exactly the same. For example, let's say you flipped two coins and one turned up heads 52 times out of 100 and one 49 times out of 100. Would you conclude the first coin is more likely to turn up heads? Of course not! But a business partner might look at an experiment, see the conversion rate is 5.4% in the control and 5.6% in the treatment, and declare the treatment a success. The decision scientist is there to help interpret the data, enforce best practices around designing experiments, **AND MORE.**

Short rule: a decision scientist uses analyses that produce *recommendations.*

1.2.4 Related jobs

While those three areas are the main types of data science positions, there are a few other distinct roles that you may see that fall outside those types. We list those jobs here since it's good to understand the positions out there and you may need to collaborate with colleagues in these positions. That said, if you are interested in one of these roles, the material in this book may be less relevant to you.

BUSINESS INTELLIGENCE ANALYST

A business intelligence analyst is a person who does work similar to that of a decision scientist, however they generally use less statistical and programming expertise. Their tool of

choice may be Excel instead of Python, and they may not ever make statistical models. While they function in role similar to a decision scientist, because of the limitations of their tools and techniques they create less sophisticated output.

If you want to do machine learning, programming, or apply statistical methods, a business intelligence analyst position could be a very frustrating since it won't help you gain new skills. They will also usually pay less than data science jobs and are considered less prestigious. But it can be a good entry point into becoming a data scientist, especially if you haven't worked with data before in a business setting. If you want to start as a business intelligence analyst and grow into becoming a data scientist, look for positions where you can learn some skills you may not have, such as programming in R or Python.

DATA ENGINEER

A data engineer is a person who focuses on keeping data maintained in databases and ensuring that people can get the data they need. They don't run reports, make analyses or develop models; instead, they keep the data neatly stored and formatted in well-structured databases so other people can. For example, a data engineer may be tasked with maintaining all of the customer records in a large-scale cloud database and adding new tables to that database as requested. Data engineers are pretty different from data scientists and they're even more rare and in demand. A data engineer may help build the data backend components of a company's internal experimentation system and update the data processing flow when the jobs start taking too long. Other data engineers develop and monitor batch and streaming environments, managing data from collection to processing to data storage. If you're interested in data engineering, you'll need strong computer science skills; many data engineers are former software engineers.

RESEARCH SCIENTIST

A research scientist develops and implements new tools, algorithms, and methodologies, often to be used by other data scientists within the company. These types of positions will almost always require PhDs, usually in computer science, statistics, a quantitative social science, or related field. Research scientists may spend weeks researching and trying out methods to increase the power of online experiments, getting one percent more accuracy on image recognition in self-driving cars, or building a new deep learning algorithm. They may even spend time writing research papers that may rarely be used within the company but help raise the prestige of the company and hopefully advance the field. Because these positions require very specific backgrounds, we will not be focusing on them in this book.

1.3 Choosing your path

In chapter 3, we'll cover some options for obtaining the data science skills, the benefits and drawbacks for each, and some suggestions for choosing between them. It's good to start reflecting here on what area of data science you want to specialize in. Where do you already

have experience in? We've seen data scientists who are former engineers, psychology professors, marketing managers, statistics students, and social workers. A lot of times, the knowledge you've gained in other jobs and academic areas can help you be a better data scientist. If you're already in data science, it's helpful to reflect now on which part of the triangle you're in. Are you happy with it? Do you want to switch to a different type of data science job? Transitioning is often available.

Can anyone become a data scientist? – Vicki Boykis

With all the optimism (and big potential salaries listed in news articles) around the data science sector, it's easy to see why it presents attractive career opportunities, particularly as the range and scope of data science job titles continues to expand. But, as a new entrant to the field, it's important to have a realistic, nuanced view of where the data science market has been headed for the next couple years and adjust accordingly.

There's a couple of trends impacting the data science field today. First, data science as a field has been around for ten years, and as such, has moved through the early stages of the hype cycle: mass media hype, early adoption, and consolidation. It's been overhyped, talked about in the media, adopted by Silicon Valley companies and beyond, and we're now at the point of high-growth adoption across larger companies, the standardization of data science workflow toolsets like Spark and AutoML.

Second, as a result, there is an oversupply of new data scientists, who've come from bootcamps, newly-established data science programs in universities, or online courses. The number of candidates per any given data science position, particularly at the entry level, has grown from 20 or so per slot to 100 or more. 500 resumes per open is not that uncommon anymore.

Third, the standardization of toolsets and the ready supply of labor, as well as the demand for people who have more experience in the field, has meant a shift in the way data science titles are distributed and a creation of a hierarchy of data science jobs and descriptions. For example, in some companies "data scientist" may mean creating models, but in some it means mostly running SQL analyses, the equivalent of what the data analyst title used to be.

This means several things for those looking to get into data science as newcomers. First and most importantly, they may find the job market to be extremely competitive and crowded, especially for those who are new to industry in general (like college graduates), or those making the transition from other industries, and competing with thousands of candidates just like them. Second, they may be applying for jobs that are not truly reflective of data science as it's portrayed in blog posts and the popular press – solely writing and implementing algorithms.

Given these trends, it's important to understand that it may be hard to initially differentiate yourself from the other stack of resumes on the pile to get into the final round of interviews. Though the strategies you read in this book may seem like a lot of work, they will help you stand out which is needed in this new, competitive data science environment.

1.4 Interview with Robert Chang, Data Scientist at Airbnb

Robert Chang is a data scientist at Airbnb where he works on the Airbnb Plus product. He previously worked at Twitter, where he worked on the Growth team doing product analytics, creating data pipelines, running experiments, and creating models. You can find his blog posts on data engineering, advice for new and aspiring data scientists, and his work at Airbnb and Twitter on Medium.com/@rchang.

1.4.1 What was your first data science journey?

My first job was as a data scientist at the Washington Post. Back in 2012, I was ready to leave academia and go into industry, but I didn't know what I wanted to do. I was hoping to be a data visualization scientist, having been impressed by work at the New York Times. When I went to my school's career fair and saw the Washington Post was hiring, naïve as I was, I just assumed they must be doing similar things to the New York Times. I applied and got the job, not doing any more due diligence.

If you were to ask for an example of how *not* to start your data science career, I definitely would volunteer myself! I got the job hoping to do either data visualization or modeling, but I realized very quickly my job was more that of a data engineer. A lot of my work was building ETL (extract transform load) pipelines, rerunning SQL scripts, and trying to make sure reports ran so we could report top-level metrics to executives. This was very painful at the time-I realized what I wanted to do was not aligned with what the company really needed and eventually left the job.

But in my subsequent years at Twitter and Airbnb, I realized I was seeing the norm and not the exception. When you're building out the data capabilities, you have to build it layer upon layer. Monica Rogati has the famous blog post on the hierarchy of data science needs, which is extremely spot-on. But at the time, I was too new to appreciate how real-live data science work was done.

1.4.2 What should people look for in a data science job?

If you're looking for a data science position, you should focus on the state of data infrastructure of the company. If you join a company where there's just a bunch of raw data that's not stored in a data warehouse, it will probably take you months or sometimes even years to get to a point where you can do interesting analytics, experimentation, or machine learning. If that's not something you expect to do, you will have a fundamental misalignment between the stage of the company and how you want to contribute to the organization.

To assess this, you can ask questions like, "Do you have a data infrastructure team? How long have they been around? What is the data stack? Do you have a data engineering team? How do they work with data scientists? When you're building out a new product, do you have a process for instrumenting the logs, building out the data tables, and putting them in your data warehouse?" If they aren't there, you will be part of the team that is responsible for building that, and you should expect to be spending quite a lot of time on that.

The second thing to look for is the people. There are three kinds of people you should be paying attention to. Assuming you don't want to be the first data scientist, you want to join a data science organization where there is an experienced leader. An experienced leader knows how to build and maintain a good infrastructure and workflow for data scientists to be productive. Second, look for a manager who is supportive of continuous learning. Lastly, it's super important, especially when you're new to work, to work with a tech lead or senior data

scientist who is very hands-on. Because for your day to day work, that's who would be the person who helps you the most.

1.4.3 What skills do you need to be a data scientist?

I think it depends on what kind of job you're looking for and what bar the employer sets. Top-tier companies in generally have a high bar, sometimes unreasonably high, because there are a lot of people trying to get in the company. They're generally looking for unicorns – someone who has data wrangling skills with R or Python as well as experience building ETL pipelines, data engineering, experiment design, and building models and putting them into production. That puts a lot of pressure on the candidates! While those are all skills you can eventually learn and may be useful for whatever problems that you're solving, I don't think they're necessary to get into data science.

If you know R or Python and a little bit of SQL, you're already in a pretty good position to get into data science. If you can plan out your career by learning more things up front, that's always helpful, but I don't think that's a requirement. It's more important to have a curiosity for learning. If you're trying to get hired by top tech companies, you need a little more, but that's more for signaling effect than what you really need on the job. It's helpful to make distinction between the core skills you need to start your career in data science and others that are nice-to-have if you want to get into a competitive, brand-name company.

1.5 Summary

- The data science skillset varies across people and positions; while some knowledge is fundamental, data scientists don't need to be experts in every relevant field.
- Data science jobs have different areas of focus: putting the right, cleaned data in front of stakeholders (analytics); putting machine learning models in production (machine learning); and utilizing data to make a decision (decision science).

2

Data science companies

This chapter covers

- The types of companies hiring data scientists
- The pros and cons of each company type
- The tech stacks you may see at different jobs

As discussed in chapter 1, Data science is a wide field with lots of different roles: research scientist, machine learning engineer, business intelligence analyst, and more. While the work you do as a data scientist depends on your role, it is equally influenced by the company you're working at. Big company vs small, tech vs traditional industry, and young vs established can all influence project focus, supporting technology, and team culture. By understanding a few archetypes of companies, you'll be better prepared when looking at places to work, either for your first data science job or your *n*th one.

The aim of this chapter is to give you an understanding of what some typical companies are like to work at each day. We are going to present five fictional companies that hire data scientists. While none of these companies are real, they are all based on research and our own work experiences and illustrate basic principles that can be broadly applied. While no two companies are exactly alike, knowing these five archetypes should help you assess prospective employers. But while these are stereotypes based on what we've seen as trends in these industries, they are certainly not gospel. You may find a company that totally breaks the mold of what we say here—or a specific team in the company that is unlike the company itself. *Lastly, while the companies in this chapter are fake, all of the blurbs you'll see are from real data scientists working at real companies!*

2.1 MTC – the Massive Tech Company

- Similar to: Google, Facebook, Microsoft
- Company history: 20 years
- Employees: 80,000

MTC is a tech company with a massive footprint, selling cloud services, consumer productivity software like a text editor, server hardware, and countless one-off business solutions. The company has amassed a large fortune and uses it to fund unusual R&D projects like self-driving scooters VR technology. While their R&D makes the news, most of the technical workforce are engineers making incremental improvements to their existing products, adding more features, improving the user interface, and launching new versions.

2.1.1 Your team: one of many in MTC

MTC has nearly a thousand data scientists spread across the company. These data scientists are largely grouped into teams each supporting a different product or division, or individually placed within a non-data science team to fully support it. For example, there are VR headset data scientists on one team, marketing data scientists on a second team, and VR-headset marketing data scientists on a third team, while the VR-headset supply chain team has one data scientist within it.

If you were a member of one of those data science teams, when you joined you would be quickly onboarded. Large organizations hire new people every day, so the company should have standard processes for getting you a laptop, access to data, and training you on how to use any special tools. On the team, you'd be tasked with doing data science for your particular area of focus. That could include creating reports and charts that executives could use to justify funding the project. It could also be building machine learning models that would be handed off to software developers to put into production.

Your team is likely to be to be large and full of experienced people. With MTC being a large, successful tech company, they have the broad footprint to draw in many good recruits to hire. Because your team is large, people within the team may be working on nearly unrelated tasks—one person could be doing an exploratory analysis for a boss in R while

another builds a machine learning model in Python for a sister team. The size of the team is a blessing and a curse: you have a large body of expert data scientists to discuss ideas with, but most of them probably don't have familiarity with the particular tasks you are working on. There is an established hierarchy on your team. The people with the more senior positions tend to be listened to more, both because they have more experience in the field and more experience with dealing with different departments at MTC.

The work your team does is likely a healthy balance of keeping the company running, such as making monthly reports and providing quarterly machine learning model updates, and doing new projects, like creating a forecast that has never been done before. The team's manager has to balance the flood of requests for data science work from other teams, which help those teams in the short term, with the desire to do innovative but unrequested work that may provide long-term benefits. With MTC's large cash stores the company can afford to do a lot more innovation and R&D than other companies, which trickles down into a willingness to try interesting new data science projects.

2.1.2 The tech: advanced, but siloed across the company

MTC is a massive organization, and with organizations of this size it's impossible to avoid using different types of technology throughout the company. One department may store order and customer data in a Microsoft SQL Server database, while a different department keeps records in Apache Hive. Worse, not only is the technology to store data disjointed, the data itself may be as well. One department may keep customer records indexed by phone number, while a different department may use email addresses to index customers.

Most MTC sized companies have their own homemade technology stacks. Thus, as a data scientist at MTC you will have to learn the specific ways to query and use data that are particular to MTC. Learning these specialized tools is great for getting you more access within MTC, but the knowledge you gain can't be transferred to other companies.

As a data scientist, you will likely have a number of possible tools you can use. Being a large organization, there will be plenty of support for major languages like R and Python since many people use them. Some teams may also use paid languages like SAS or SPSS, but this is a bit rarer. If you want to use an unusual language that you enjoy but few other people use, such has Haskell, you may or may not be able to, depending on your manager.

The machine learning stack varies dramatically by what part of the company you are in. Some teams use microservices and containers to efficiently deploy models, while others have antiquated production systems. The diversity in tech stack for deploying software makes it difficult to connect to other team's APIs—there is no one central location to learn about and understand what is going on.

2.1.3 The pros and cons of MTC

Being a data scientist at MTC means having an impressive job at an impressive company. Because MTC is a tech company, people know what a data scientist is and what helpful things

you can do. Having a universal understanding of your role makes the job a lot easier and is a huge perk. The high number of data scientists in the company means you have a large support network you can rely on if you are struggling, and smooth processes for joining the company and getting access to required resources. It'll be rare for you to find yourself stuck and on your own.

Having lots of data scientists around you comes with cons as well. The tech stack is complex and difficult to navigate because so many people have built it up in so many ways. An analysis you've been asked to recreate may be written in a language you don't know by a person no longer around. It'll be harder to stand out and be noticed, because there are so many other data scientists around you. And you may find it difficult to find an interesting project to work on, because so many of the obvious projects have already been started by other people.

With MTC being an established company, working there gives you more job security. While there is always the risk of layoffs, it's not like working at a startup where funding could dry up at any moment. And at large companies, managers lean more towards finding a new team for someone to work on rather than firing them—firing opens up all sorts of legal complications that require thorough back-up support for the termination decision.

Something that is both a pro and con of MTC is that there are people in many specialized roles within the company. There are data engineers, data architects, data scientists, market researchers, and more, all doing different roles that relate to data science. This means you'll have lots of people to pass work off to— for example, you have a low chance of being forced to create your own database. This is great for passing off work you outside your expertise, but it also means you can't stretch your skills.

Another con of MTC is the bureaucracy. Being a large company means that getting approvals for things like new technology, trips to conferences, or starting projects can require going far up the chain of command. Worse, the project you've been working on for years could be cancelled because two executives are fighting, and your project is collateral damage.

MTC is a great company to work at for data scientists who are looking to help solve big problems using cutting-edge techniques—both for decision scientists wanting to do analyses and machine learning engineers wanting to build and deploy models. Large companies have lots of problems to solve, and a budget that allows for trying out new things. You may not be able to make big decisions yourself, but you'll know you've contributed. MTC is a poor choice for a data scientist who wants to be the decision maker and call the shots—the large company has established methods, protocols, and structures that you have to fall into.

2.2 HandbagLOVE – the established retailer

- Similar to: Payless, Bed Bath & Beyond, Best Buy
- Company history: 45 years
- Size: 15,000 employees (10,000 in retail stores, 5,000 in corporate)

HandbagLOVE is a retail chain with 250 locations across the US, all selling purses and clutches. The company has been around for a long time and is filled with experts in how to lay out a store and improve the customer experience. The company has been generally slow to adopt new technology, taking plenty of time before getting its first website and its first app.

Recently, HandbagLOVE has been seeing sales drop as Amazon and other online retailers have eaten away at its market share. Knowing that the writing is on the wall, HandbagLOVE has been looking to improve via technology, investing in an online app, an Amazon Alexa skill, and trying to use the value they have in their data. HandbagLOVE has had financial analysts employed for many years calculating high-level aggregate statistics on their orders and customers; however, only recently have they considered hiring data scientists to better understand customer behavior.

The newly formed data science team was built starting with a base of financial analysts who had previously been making Excel reports on performance metrics for the company. As HandbagLOVE supplemented these people with trained data scientists, the team started to provide more sophisticated products: monthly statistical forecasts on customer growth in R, interactive dashboards allowing executives to better understand sales, and a customer segmentation that buckets customers into helpful groups for marketing.

While the team has made machine learning models to power the new reports and analyses, HandbagLOVE is far from deploying machine learning models into continuously running production. Any product recommendations on their website and app are powered by 3rd party machine learning products, rather than having been built within the company. There is talk on the data science team of changing this, but who knows how many years away it is.

2.2.1 Your team: a small group struggling to grow

The team leans heavily towards data scientists who can do reporting rather than being trained in machine learning—a topic very new to many. That newness can be extremely helpful—by being new, they will likely be very interested in learning more and trying new things. It can

also prove difficult: some of their programming and statistical methods may be inefficient or even wrong since they haven't been taught the correct way to do things.

HandbagLOVE has laid out general paths to progress into senior roles. Unfortunately, none of them are specific to data science—they are just high-level goals around being independent or things taken from the software development career management tools which don't quite apply. To progress in your career, you mostly have to convince your manager that you're ready, and hopefully your manager can get approval to promote you. On the plus side, if the team ends up growing you'll quickly become a senior person on the team.

Since the data science team provides reports and models for departments throughout the company, like marketing, supply chain, and customer care, the members of the data science team are well known. This has given the team a great deal of respect within the company, and in turn the data science team has a lot of comradery within it. The combination of the size of the team and the level of influence within the company allows the data scientist to have far more influence than they would in other companies. It's not unusual for someone on the data science team to meet with top level executives and contribute to the conversation.

2.2.2 Your tech: a legacy stack that's starting to change

The common phrase you hear when talking about technology at HandbagLOVE is "well that's how it's always been." Order and customer data are stored in an Oracle database that is directly connected to the cash register technology and hasn't changed in 20 years. The system has been pushed well past its limits and has had many modifications bolted on. All that being said, it still works. Other data is collected and stored in the central database as well: data collected from the website, data from the customer care calls, and data from promotions and marketing emails. All of these servers live on-prem (not on the cloud), and an IT team keeps them maintained.

By having all of the data stored in one large server, you have the freedom to connect and join the data however you want. And while sometimes your queries may take forever or overload the system, usually you can find a workaround to get you something usable. The vast majority of analyses are done on your laptop. If you need a more powerful computer to train a model it's a large hassle to get it.The company doesn't have a machine learning tech stack because they don't have any in-house machine learning.

2.2.3 The pros and cons of HandbagLOVE

By being at HandbagLOVE, you have a lot of influence and ability to do what you think is wise. You can go from proposing making a customer lifetime value model, building it, and using it within the company without having too many people you have to convince to let you run with your idea. This is due to a combination of the size of the company and the newness of data science. This freedom is very rewarding; you are incredibly empowered to do what you think is best.

The downside of this power is that you don't have many people to call on for help. If you try and make a customer lifetime value model and the model just isn't performing well, you have few people to support you. You're responsible for finding a way to make things work or dealing with the fallout when things don't work.

The tech stack is antiquated, and you'll have to spend a lot of time making workarounds for that—which is a hassle. You may want to use a newer technology for storing data or running models, but you won't have the technical support to do it. If you're not able to entirely set up any new technology yourself you'll just have to get by without using it.

A data scientist's salary won't be as high as it would be at bigger companies, especially tech ones. HandbagLOVE just doesn't have the cash available to pay high data science salaries—and besides the company doesn't need the best of the best anyway, just people who can do the basics. That being said, the salary won't be terrible; it'll certainly be well above what most people at the company make with similar years of experience.

HandbagLOVE is a good company to work at for data scientists who are excited to have the freedom to do what they think is right—but perhaps aren't interested in using the most state-of-the-art methods. If you're comfortable using standard statistical methods and making more mundane reporting, HandbagLOVE should be a comfortable place to grow your career. If you're really only interested in using start-of-the-art machine learning methods then you won't find many projects to do at HandbagLOVE, nor will you find many people there who know anything about what you're talking about.

2.3 Seg-Metra – the early-stage startup

- Similar to: a thousand failed startups you haven't heard of
- Company history: 3 years
- Size: 50 employees

Seg-Metra is a young company that sells a product that helps client companies optimize their website by customizing for unique "segments" of customers. Seg-Metra sells its product to

businesses, not consumers. Early on in its brief history, Seg-Metra got a few big-name clients to start using the tool, which helped the company get more funding from venture capitalists. Now with millions of dollars at hand, the company is looking to quickly scale in size and improve the product.

The biggest improvement the founders have been pitching to investors is to start adding basic machine learning methods to the product—it was pitched to investors as "cutting-edge AI." With this new funding in hand, the founders are looking for machine learning engineers to build what they have pitched. There is also a need for decision scientists to start reporting on the usage of the tool, allowing the company to better understand what improvements to make in the product.

2.3.1 Your team—what team?

Depending on when a data scientist gets hired, they may very well be the first data scientist in the company. If not the first, they will be in the first few data science hires and likely report to the one who was first hired. Due to the newness of the team there will be few to no protocols—no established programming languages, best practices, ways of storing code, or formal meetings.

Any direction will come from that first data scientist hire, who will be in charge of starting to establish protocols. The culture of the team will likely be set by their benevolence. If that person is open to group discussion and trusting of the other team members, then the data science team as a whole will decide things like what language to use. If that person is controlling and not open to listening, they will make these decisions themselves.

Such an unstructured environment can create immense comradery. The whole data science team works hard and struggles to get new technologies, methods, and tools all working and can form deep bonds and friendships. Alternatively, immense emotional abuse could happen at the hands of those with power, and since the company is so small there is little accountability. Regardless of exactly how Seg-Metra's growth shakes out, the data scientists at this early stage company are in for a bumpy and wild ride.

The work of the team can be fascinating or frustrating depending on the day. Oftentimes data scientists are doing analyses for the first time ever—like making the first attempt at using customer purchase data to segment customers or deploying the first neural network into production. These first-time analyses and engineering tasks are fascinating because they are uncharted territory within the company, and the data scientists get to be the pioneers. On other days, the work can be grueling—like if a demo has to be ready for an investor and the model is still not converging the day before. And even if the company has data the infrastructure may be so disorganized that the data can't be feasibly used. While the work is chaotic, all of these tasks mean that the data scientists learn lots of skills very quickly while working at Seg-Metra.

2.3.2 The tech: cutting edge technology that is taped-together

By being a young company, Seg-Metra is not constrained by having to maintain old legacy technology. Seg-Metra also wants to impress its investors, which is a lot easier to do when your technology stack is impressive. Thus, Seg-Metra is powered by the most recent and greatest methods of software development, data storage and collection, and analysis and reporting. Data is stored in an assortment of modern cloud technologies, and nothing is done on-premise. The data scientists connect directly to these databases and build machine learning neural network models on large AWS virtual machines instances with GPU processing. These models are deployed using modern software engineering methods.

At first glance, the tech stack is certainly impressive. But the company is so young and growing so fast that there are constantly issues with the different technologies working together. When the data scientists suddenly notice missing data in the cloud storage, they have to wait for the overworked data engineer to fix it (and that's if they're lucky enough to have a data engineer). It would be great if Seg-Metra had a dedicated DevOps team to help keep everything running, but so far the budget has been spent elsewhere. Further, the technology has all been installed so quickly that even though the company is young it would be difficult to monitor it all.

2.3.3 Pros and cons of Seg-Metra

As a growing startup there is a lot of appeal to working at Seg-Metra. The growth of the company is providing all sorts of interesting data science work and an environment where you are forced to learn quickly. These sorts of positions can teach skills that jump start a career in data science: skills like working under deadlines with limited constraints, communicating effectively with non-data scientists, and knowing when to pursue a project or decide it's not worth it. Especially early in a career, developing these skills can make you much more attractive as an employee than people who have only worked at larger companies.

Another pro of working at Seg-Metra is that you get to work with the latest technologies. By using the latest tech your job *should* be more enjoyable: presumably the new technologies coming out are better than the old technologies. By learning the latest tech, you should also have a more impressive resume for future jobs. Companies looking to use newer technology will want you to help guide the way.

While the pay is not as competitive as larger companies, especially tech companies, the job does provide stock options which have the potential to be enormously valuable. If the company eventually goes public or gets sold, those options could be worth hundreds of thousands of dollars or more. Unfortunately, the odds of that happening are somewhere between "getting elected to city council" and "getting elected to US Congress." So this is only a pro if you enjoy gambling.

One con of working at Seg-Metra is that you have to work very hard. Having 50- to 60-hour work weeks is not uncommon, and the company expects everyone to contribute everything they can. In the eyes of the company, if everyone isn't working together the

company won't succeed, so are you really going to be the one person to use all of their vacation time in a year? This can create a hugely toxic environment ripe for abuse and a lot of employee burnout.

The company is volatile, relying on finding new clients and help from investors to stay afloat, giving Seg-Metra the con of low job security. It's possible that in any year the company could decide to lay off people or go under entirely. These changes can happen suddenly without warning. Job insecurity is especially difficult for people with families so this causes the demographics of the company to skew younger. A young workforce can also be a con if you want to work with a more diverse, experienced team.

Overall, working at Seg-Metra provides a great opportunity to work with interesting technology, learn a lot quickly, and have a small chance of making a ton of money. But to do so requires an immense amount of work and potentially a toxic environment. So this company is best for data scientists looking to gain experience and then move on.

Rodrigo Fuentealba Cartes, Lead Data Scientist at a small government consulting company

The company I work at provides analytics, data science and mobile solutions for governmental institutions, armed and law enforcement forces, and some private customers. I am the Lead Data Scientist, and I am the only one in charge of data science projects in the entire company. We don't have data engineers, data wranglers or any other data science roles there, because the department is relatively new. Instead, we have database administrators, software developers, systems integrators, and I double as a system/software architect and open source developer. That might look odd, and definitely puts strain on me, but it works surprisingly well.

One strange story from my job: I was working in a project that involved using historical information from many environmental variables such as daily weather conditions. There was a lack of critically needed data because an area of study didn't have weather stations installed. The project was in jeopardy and the customer decided to shut the project down in a week if their people could not find the information.

I decided to fly to the area and interview some fishermen and asked how they knew that it was safe to sail. They said they usually sent a ship that transmitted the weather conditions over the radio. We visited a radio station, and they had handwritten transcripts of communications since 1974. I implemented an algorithm that could recognize handwritten notes and extract meaningful information, and then implemented a natural language processing pipeline that could analyze the strings. Thanks to going out to the field and finding this unusual data, the project was saved.

Gustavo Coelho, a data science lead at a small start-up

I have been working for the last 11 months in a relatively new startup which focuses on applying AI to HR management. We predict future performance of candidates or their likelihood of being hired by a certain company. Those predictions are aimed at helping speed up the hiring process. We rely heavily on bias mitigation in our models. It's a small company, we have 11 people, and the Data Science team makes up 5 of them including me. The whole of the company is dedicated to helping the Data Science team deliver the trained models into production.

Working at a small startup gives me the chance of learning new concepts and applying them every day. I love thinking about the best way to setup our data science processes so we can scale and give more freedom to our data scientists to focus on data science. HR is not a tech savvy field, so more than half of the project length is spent explaining the

solution to our clients and helping them get comfortable with the new concepts. And then when we finally get the go ahead there is also a lot of time spent coordinating with the client's IT department to integrate into our data pipeline.

2.4 Videory – the late-stage, successful tech start-up

- Similar to: Lyft, Twitter, and Airbnb
- Company history: 8 years
- Size: 2,000 people

Videory is a late-stage, successful tech start-up that runs a video-based social network. Users can upload 20 second videos and share them with the public. They have just gone public and everyone is ecstatic about it. They're not close to the size of MTC, but they're doing well as a social network and growing the customer base each year. They're data-savvy and have probably had data analysts or scientists for a few years now or even since the start. The data scientists on the team are very busy doing analyses and reporting to support the business, as well as creating machine-learning models to help pair people with artists to commission work.

2.4.1 The team: specialized but still room to move around

Videory is still at the point where you can gather all the data scientists in an extra-large conference room, or at least a small auditorium. Given the size of the company, the team may be organized into a *centralized model*. Every data science person reports to a data science manager, and they all are in a single large department of the organization. They help other teams throughout the company, but ultimately they set their own priorities, and some data scientists are working on long-term projects that have no immediate benefits.

There's specialization among the data science team at Videory given the size of the company. The company has some delineation between people who do the heavy machine learning, statistics, or analytics. Videory is small enough that it's possible to switch between these groups over time. There will usually be some interaction between the all the data scientists – training sessions, monthly meetings, a shared slack channel – that you wouldn't find in MTC where there are hundreds or thousands of data scientists. The different sub-teams likely use different tools, and there is a group of people with PhDs who publish academic papers and do more theoretical work.

2.4.2 The tech: trying to not get bogged down by legacy code

Videory has a lot of legacy code and technology, and probably at least a few tools they developed internally. Videory is likely trying to keep up with tech developments and has plans to either switch over to a new system or supplement their existing ones with new technologies. Like most companies, a data scientist there will almost definitely query a SQL database to get data. You also will probably have some business intelligence tools as well since there's a lot of non-data science consumers.

As a data scientist at Videory you'll definitely get to learn something new. All these companies have big data and systems to deal with it. SQL won't be enough – the company has billions of events they need to process a month. You may be able to try out Hadoop or Spark when you need to pull out some custom data that's not stored in the SQL database.

The data science is typically done in R or Python, with plenty of experts available to provide assistance if things are proving difficult. The machine learning is deployed through modern software development practices like using microservices. Because of how well-known the company is for being a successful startup, lots of talented people are working at the company and bringing their cutting-edge approaches.

2.4.3 The pros and cons of Videory

Videory can be a good size for data scientists – there are enough other data scientists around for mentorship and support, but the team is still small enough that you can get to know everyone. Data science is recognized as important on the company-level, which means your work can get recognition from vice presidents and maybe even the C-suite (CEO, CTO, etc.). You'll have data engineers to support your work. The data pipelines may get slow sometimes or even rarely break, but you won't be responsible for fixing it.

In a 1000+ person organization, you will need to deal with inevitable political issues. You may be pressured to generate numbers that match what people want to hear (and can tell their bosses to get a bonus) or face unrealistic expectations of how fast something can be developed. You can also end up working on things that aren't really needed by the business because your manager asked you to. You'll sometimes end up feeling like you have had no direction or your time was wasted. While not as much as at an early-stage startup, the organization will still be changing a lot – what's a priority one quarter can be totally ignored the next.

While other data scientists at Videory will be more knowledgeable than you on most data science topics, you might quickly become the expert on a specific one, like time series analysis. This can be great if you like mentoring and teaching others, especially if your work supports you taking time to learn more on your particular field of expertise by reading papers or taking courses. But it can be hard when you feel like there's no one who could check your work or push you to learn new things. While you'll always have more to learn, it may not end up being the area you want to focus on.

Overall, Videory provides a nice blend of some of the benefits from the other archetypes — large enough that there are people around to provide help and assistance when needed, but not so large that requests get stuck in bureaucratic madness or that departments overlap in scope. Data scientists working at the company get plenty of chances to learn, but due to the specialization of roles they don't quite get the opportunities to try everything. This company is a great place for data scientists looking for a safe bet that provides chances to grow, but not so many that it's overwhelming.

Emily Bartha, the first data scientist at a mid-sized startup

I am the first data scientist at a mid-sized startup that has a product focused on insurance. As the first data scientist, I get to help define our strategy around using data and introducing machine learning into our product. I sit on the Data Team in the company, so I work very closely with data engineers, as well as our Data Product Manager.

A day in my life at work starts with morning standup with the Data Team. We talk about what we have planned for the day and any blockers or dependencies. I spend a lot of time digging through data - visualizing, creating reports, and investigating quality issues or quirks in the data. I spend a lot of time on documentation too. When I code, I use GitHub like the rest of the engineering team and have team members review my code (and I review theirs). I also spend a good chunk of the day in meetings, or side-of-desk collaboration with members of my team.

Having worked at bigger companies in the past, I love working at a small company! There is a lot of freedom to take initiative here - if you have an idea and want to work to make it a reality, no one will get in your way. Look for a company that has already made an investment in data engineering—when I arrived there were already several data engineers and a strategy for instrumentation, data collection, and storage. When you work at a small company, things are constantly changing and priorities are shifting, which makes it important to be adaptable. People who enjoy diving deep on a project and working on it for months may not enjoy working at a startup because it often requires developing solutions that are "good enough" and moving on to the next thing.

2.5 Global Aerospace Dynamics (GAD) – the massive government contractor

GLOBAL AEROSPACE DYNAMICS

- Similar to: Boeing, Raytheon, Lockheed Martin
- Company history: 50 years
- Size: 150,000 people

Global Aerospace Dynamics is a massive and rich company, bringing in tens of billions of dollars in revenue each year through various government contracts. The company develops everything from fighter jets and missiles to intelligent traffic light systems. The company is spread across the country through various divisions, most of which don't talk to each to each

other. GAD has been around for decades, and many people working there have been there for decades too.

GAD has been slow on the uptake of data science. Most of the engineering divisions have been collecting data, but they struggle to understand how it can be used in their very regimented existing processes. Because of the nature of the work, code needs to be extremely unlikely to have bugs and ruthlessly tested, so the idea of implementing a machine-learning model, which has limited predictability when live, is dicey at best. In general, the pace of work at the company is slow; the tech world "move fast and break things" is the polar opposite of the mentality at GAD.

With the number of articles on artificial intelligence, the rise of machine learning, and the need for using data to transform a business, the executives in GAD are all ready to start hiring data scientists. Data scientists are showing up on teams throughout the organization, with tasks like analyzing engineering data for better reporting, building machine learning models to put into products, and working as service providers to help GAD customers troubleshoot problems.

2.5.1 The team—a data scientist in a sea of engineers

While it depends on where in GAD you are and what project you are working on, most data scientists are a single person on a team of engineers. At best there may be two or three data scientists on your team. The data scientist has the job of supporting the engineers with analysis, model building, and product delivery. Most of the engineers on the team only have a very loose understanding of what data science is—they remember regressions from college but don't know the basics around collecting data, feature engineering, the difficulties of validating a model, or how models get deployed. You will have few resources to help you when things go wrong, but since so few people understand your job no one else might notice that things are going wrong.

Many of the engineers on the team will have been with the company for ten or more years, so they will have plenty of institutional knowledge. They'll also be more likely to have a mindset of: "we've been doing things this way since I've been here, why should we change?" That attitude will make it more difficult for ideas proposed by data scientists to actually be implemented. The slower nature of the defense industry means that people tend to work less hard than in other places—people clock in 40 hours a week but casually slipping down below that isn't unusual. At other companies it can be stressful to be overwhelmed by too many tasks, while at GAD the stress comes from not having enough work to do and being bored.

Promotions and raises are extremely formulaic, both due to following rules to reduce bias (and thus less likely to get GAD sued) and because that's how things have been done for decades. Getting raises and promotions largely has to do with how many years you've worked at the company. Being an extremely hard worker may make your next promotion come a year earlier or earn you a somewhat higher bonus, but there is little chance of a junior data scientist quickly rising to a lead data scientist. The flip side of this is that people getting fired is rare.

2.5.2 The tech: old, hardened, and on security lockdown

While the technology stack greatly differs between groups in GAD, it all tends to be relatively old, on-premises instead of in the cloud, and covered in security protocols. Since the data involved covers topics like fighter jet performance, it's essential for the company that it isn't leaked. Further, the company needs legal accountability for any technology they use in case something goes wrong, so open source tends to be frowned upon. For example, while Microsoft SQL Server is more expensive than PostGRES SQL, GAD is happy to pay Microsoft money knowing that if there is a security bug they can call Microsoft to deal with it.

In practice, this looks like data being stored in Microsoft SQL Server databases run by an IT team that is extremely stingy with who has access to what. The data scientists are allowed to access the data; however, they have to run Python on special servers that have limited internet access, so that any libraries don't secretly send data to foreign countries. If the data scientists want to use special open source software there is little chance of IT and security approving it. This ends up making it much more difficult for the data scientists to do work.

If code needs to be deployed to production systems, it tends to be in traditional ways—GAD is just beginning to adopt modern methods of putting machine learning code into production.

2.5.3 The pros and cons of GAD

The pros of working at GAD are that the data science jobs are slow, comfortable, and secure. The less rigorous pace of the job means you're more likely to have energy left when you get home for the evening. You'll often find yourself with free time when you are working, which you can spend reading data science blogs and articles without anyone complaining. The fact that few other people know the basics of data science means that you'll have fewer people questioning you. And because it's a massive organization worried about legal liabilities, you would have to really underperform before you're fired.

The downsides of working at GAD are that you are less likely to learn new skills than you would at other companies. You'll likely be assigned to a single project for years, and so the technologies and tools for it will quickly become mundane. Worse, the skills you do learn will be for outdated technology that isn't transferrable to other institutions. And while you won't easily get fired, you also won't easily get promoted.

GAD is a great place to work if you find a team doing projects you find enjoyable and you don't want work to be your life. Many people work for GAD for decades because it's comfortable and they are happy with that. But if you are a person who demands challenges to keep you going, GAD might not be a good fit.

Nathan Moore, a Data Analytics Manager for a utilities company

The company I work at provides and sells power for hundreds of thousands of people, and the company is partially owned by the government. The company itself has around 1,000 employees spread across many different functions. My job involves investigating and prototyping new data sources and working with the database specialists to clean and

document current data sources. We've got a bunch of legacy systems and new initiatives happening so there's always something to do.

At the moment a day in the life involves meetings, reviewing specifications for ETL, trying out a new machine learning technique I found on Twitter, giving feedback on reporting, learning to use JIRA and Confluence, and many emails. In the past I've been involved in model development and assessment, data analysis when some overnight processing fails, and submissions to government on an industry-wide review of the sector.

We're large enough that we've got a good team of analysts to work on a variety of projects, from day to day reporting to large customer segmentation projects. I've had lots of opportunities to move around in the business and have worked here for 11 years. But since we have billions of dollars of assets risk aversion is high within the company and pace of change is a little bit slow. We have a large enough IT department that can support everyday functions, but any significant project like the systems upgrade means resources are scarce for any non-priority improvements. Everything needs to be justified and budget set aside and there is plenty of politics to navigate.

2.6 Putting it all together

We can summarize the five companies here into a few key ideas:

- MTC – a massive tech company with lots to learn from, but a wildly complex tech stack and plenty of institutional chaos
- HandbagLOVE – a mid-size retailer that doesn't have much data science, but provides lots of opportunities to try new things
- Seg-Metra – a startup that is growing and needs data scientists, but hasn't figured out quite what to do and burns plenty of employees out
- Videory – a mid-size tech company that has a team of data scientists who have specialized, but isn't overwhelmed by bureaucracy
- GAD – a defense contractor that is hiring lots of data scientists to do work that isn't particularly demanding

When looking for companies to work at, you'll find that many of them are similar to these companies in various ways. It can be helpful as you go through job applications and interviews to try and understand what the strengths and weaknesses of working at these different companies are.

A summary of the different companies that hire data scientists

Criteria	MTC	HandbagLOVE	Seg-Metra	Videory	GAD
	Massive tech	Retailer	Startup	Mid-tech	Defense
Bureaucracy	Lots	Little	None	Some	Lots
Tech stack	Complex	Old	Fragile	Mixed	Ancient
Freedom	Little	Lots	TONS	Lots	None
Salary	Amazing	Decent	Poor	Great	Decent

Job security	Great	Decent	Poor	Decent	Great
Chances to learn	Lots	Some	Lots	Lots	Few

2.7 Interview with Randy Au, Quantitative User Experience Researcher at Google

Randy Au works on the Google Cloud team. Having worked in data science with a focus on human behavior for over a decade, he blogs about how to think about working at start-ups and different types of companies on medium.com/@randy_au.

2.7.1 Are there big differences between large and small companies?

Yes—usually it's more organizational and structural. There are points in a company where culture changes because of the scale. At a 10-person start up everyone does everything because everyone's wearing all the hats. Meanwhile, around 20 people things start specializing. You start getting three or four-person teams dedicated to specific things. People can think harder about certain things and you don't have to learn everything about a company. At around 80 to 100 people the existing teams don't scale anymore. Then there's a lot more process around things. You don't know everyone in the company anymore. You don't know what everyone's up to, and so there's a lot more overhead to reach common understanding. Beyond that, after about 150-200 people, it's impossible to know what's going on around the company, so the bureaucracy has to exist. Then you go to Google, which is 100,000 people. There you have no idea what most of the company is doing.

The smaller the company, the more likely you're going to interact with everyone in the company. At a 40-person company I would have the CEO of the company sitting at my desk as we're both exploring a data set together. That will never happen in Google. But are you okay with the situation that happens in a lot of start-ups where you're building an F1 car and you're driving it at the same time and everyone's arguing whether you should have a steering wheel? When you're the "data person" at a small company, the methods don't really matter as much – you're just trying to squeeze all the data and get some insights out of it. It's okay to not be as rigorous so you can make decisions more quickly.

2.7.2 Are there differences based on the industry of the company?

Some industries have historically had math or data people. For example, an insurance company has actuaries. Those people have been around for a hundred years and they really know their stats. If an insurance company is going to bring in data scientists, they come with a slightly different view on it. They already have this built in structure for extremely talented stats people. They're going to be filling a gap somewhere else: there's going to be a gap in their big data or in optimizing their website or something.

Finance also has a long tradition of having quants. I remember failing a quant finance interview once because they did a code test. But as a data scientist I just make sure my code

is functional and gives the correct answer; I don't think too hard about performance until it becomes a problem. Their coding test literally tested you on performance and dings you points for not being performant automatically. I was like "oh yeah, you guys are in finance I get it."

2.7.3 Should new data scientists be wary of startups since they'll do tons of cleaning, pipeline creation, and unglamorous work?

I think if you talk to everyone who's doing data science work, the vast but silent majority are people who are doing this kind of grunt work that's not sexy at all. I got a ridiculous amount of responses to the article I wrote about data science at startups that were people saying: "Yeah, this is my life." This is not what people talk about when they talk about data science. It's not the sexy: "here's a new shiny algorithm I applied from this arXiv paper." I don't think I've applied anything in an arXiv paper in the twelve years I've worked. I'm still using regression because regression really works! I think that is the reality of it.

You're going to be cleaning up your data; I don't think there's anyone even at the Facebooks and the Googles who doesn't have to clean data. It might be slightly easier to clean up your data because there's structure around it. But no, you're going to have to clean up your data—it's a fact of life.

2.7.4 What's your final piece of advice for aspiring and junior data scientists?

Know your data. This does take a long time - six months to a year or more if it's a complicated system. But your data quality is the foundation of your universe. If you don't know your data, you're going to make a really bizarre statement about something that your data just can't let you say. For example, some people will say, "Oh, I have the number of unique cookies visiting my website, and that's equal to number of unique people." But that's not true—what about those people who are using multiple devices or browsers?

To really know your data, you need to make friends with the people with domain knowledge. For example, when I was doing financial reports, I made friends with the finance people so I could learn the conventions accounting has about how they name things and the order of how things are subtracted. Maybe you got fifty million pages from this one IP, and someone else will realize that's IBM. You won't know all this stuff, but someone probably will.

2.8 Summary

- There are many different types of companies that hire data scientists.
- The data science job itself varies, largely based on the company's industry, size, history and team culture.
- It's important to understand what kind of company you are considering working at.

3

Getting the Skills

This chapter covers

- The different methods to learn data science
- Understanding what makes a good academic program or boot camp
- Choosing the route best for you

Now that you've decided to become a data scientist, you need to acquire the skills! Fear not: wondering how to learn the skills of a data scientist is one of the universal parts of becoming a data scientist. There are so many ways to do it, from watching YouTube videos to getting a degree, and lots of people will tell you their way is the only correct path. Worse, it's easy to feel overwhelmed by the amount to learn: algorithms, programming languages, statistical methods, and then you throw in all the business demands. Just thinking about it can be emotionally draining.

The good news is that there are really only four main methods to get the skills you need. Each method has its advantages and drawbacks, but once you lay them out it usually becomes clear which is the right approach for you. By the end of this chapter you should understand the different methods, and after some reflection you should be able to decide the best route for your situation. You can do this!

The four methods for gaining data science skills that we will cover are:

1. Earning a graduate degree in data science or a related field.
2. Participating in a data science boot camp, an 8-12-week crash course in data science.
3. Doing data science work in your current job.
4. Teaching yourself through online courses and through data science books.

Let's walk through each of them.

What if you haven't graduated college?

Most of this chapter is designed with the assumption that the reader has graduated college, most likely with a technical degree. If you don't meet that criteria, don't worry most of the chapter still applies. There are a few adjustments that need to be made when reading it.

If you haven't graduated college then you'll likely want to get an undergraduate degree first before following the chapter. Your best bet is a related technical degree that will teach you some data science skills like mathematics, statistics, or computer science. If you choose one of these majors, try and use any flexible credit requirements to fill gaps in your data science skill set. Some schools now offer data science undergraduate degrees, which should make you extremely well suited to get a job after college. After finishing your degree you should be able to get a data science job straight out of school (especially if you follow the guidance of Parts 1 and 2 of this book). You can also choose to following additional steps listed in this chapter, like teaching yourself extra data science skills or doing data science work during your first job.

If you have graduated college but with a non-technical degree, then this chapter's guidelines still hold. It might be an extra good idea to get a data science graduate degree, since the longer time period of study will give you more time to get up to speed. You may be inclined to get a second, technical undergraduate degree, but avoid this at all costs. It's extremely expensive and time consuming—that knowledge can be gained in other ways.

3.1 Earning a data science degree

Many colleges now offer graduate degrees in data science, and the programs cover a mixture of topics from computer science, statistics, and the business school. Since they are master's degree programs, they generally take two years and cost up to $70k or more. As with most graduate programs, you can often choose to go through the program more slowly while still having a job and/or potentially take them as online courses. While many schools offer an actual data science degree, depending on your interests you may alternatively get a degree in computer science, statistics, business analytics, operations research, or something very close to data science.

The good thing about a data science degree is that it's exhaustive–due to the length of the program and the amount of time you spend in it, you should learn everything you need to be a junior data scientist. Through coursework and projects, you'll get experience using the methods and hands-on programming. If you come into the program without much programming experience you should be able to pick it up on the way (although you may have to take an extra course or two).

Graduate degrees to learn data science have a few downsides. First, they are extremely expensive, both in terms of the cost of tuition and the opportunity cost from not earning income and gaining direct work experience during the time you are a full-time student. Graduate programs are an order of magnitude more expensive than the other options both in money and in time. Spending years studying before you feel ready to switch careers is a huge amount of time in your life, and if you decide part-way through you don't want to be a data scientist you can't get that money or time back.

If you are coming from a background that's related to data science, like software development, or have substantial undergraduate coursework in the field, a graduate program

will teach you a lot of stuff you already know. That means from a long program you may only get a small amount of useful, new information. This can be a huge drawback and make the program feel frustrating.

Also, these programs are taught by academic professors. Many academic professors have spent their whole career in academia, making the material they teach potentially different than what people use in industry. That means that a particularly disengaged professor may do things like use old languages such as SPSS, or they won't understand modern tools like version control. This is especially common in degree programs outside of data science. Some universities bring people from industry in to teach courses, but then they may or may not know much about teaching. It's difficult to tell how much of the program uses modern techniques until you're in it. During the application process, try and find opportunities to talk to current or former students to get a feeling for the program and how useful it is for a career.

3.1.1 Choosing the school

As you start searching for graduate data science programs, you may be quickly overwhelmed by the number of options you have. Worse, you may find yourself getting your mailbox full of flyers for different programs, and calls from recruiters. Depending on how much work you want to do, you should likely apply to somewhere between three and ten of these programs. Apply to too few and it's possible you won't get into any; apply to too many and you'll find yourself devoting excessive time (and application fees) to graduate applications. To decide which schools to apply to, consider these different metrics:

- **If you'll be happy with the location and lifestyle [very important]** – you'll likely be looking at graduate programs all across the country, but what your life looks like outside of school will be quite different if you go to a school in Los Angeles versus upstate New York. If the climate, proximity to friends, and cost of living don't work for you, it doesn't matter how good the program is because you'll be unhappy doing it.
- **What topics the program's coursework covers [important]** – because data science is so new, universities may have dramatically different coursework. This is especially complicated by which department the program lives in; a CS-based data science program is going to be focused on methods and algorithms while a business school program will be focused more on applications and rely on case studies. Check that the course material covers weaknesses in your skillset from Chapter 1.
- **How much project work the program has [important]** – the more project work a program has, the more you'll learn about how data science works in practice and the better prepared you'll be for industry (projects are covered further in the next chapter). Significant projects are also great for putting on your resume, which can help you get an internship during the graduate program or help get that first job.
- **Where graduates end up [important]** – often the school will provide statistics on where students go after they graduate. These statistics can be informative, but often the schools will show metrics that make them look the best, even if they are misleading

(which ironically is a skill you learn as a data scientist). If possible, try reaching out to some of the program alumni through LinkedIn to get an unbiased perspective on how graduates fare. Especially if you want to work at an MTC, you can look at what companies recruit directly at that school. While of course you can still apply for a job if they don't, your job application may be given less consideration.

- **Funding [rare but very important]** – in rare circumstances schools will offer funding for master's graduate students. This means the school will pay for your coursework and pay a stipend, usually provided you are a teaching assistant for a class. If you get this we highly recommend you take it—not having to pay for your degree and getting a salary to boot is financially a much better option than paying yourself. If the funding involves teaching, you'll also have the benefit of being forced to learn how to communicate with a roomful of people which will be helpful in your data science career. The downside is that teaching takes lots of time, which distracts from your studies.
- **How much the program is connected with businesses in the area [medium]** – if the school does a lot of work with local companies, especially tech ones, it shows that the school is connected with the community. That'll make it easier to get an internship or a job and will give you more interesting materials during classes. It also lowers the chance of having professors who are out of touch with the methods used outside of academia.
- **Admission requirements [not very important]** – some schools have requirements around courses you need to have taken to be accepted. Most programs require that you have some courses in mathematics like linear algebra, and some courses in programming like an introduction to Java. If you are missing one or two of the courses, you may be able to get the requirement waived or take make-up courses once you're in the program. If you are finding you don't have any of the prerequisites, or they require a specific undergraduate degree like Computer Science, the program may not be a good fit for you.
- **Prestige of the school [not at all important]** – unless you get accepted to an extremely prestigious school such as Stanford or MIT, employers won't care about how well-known the school is. Prestige mainly matters if you are looking to go into academia instead of industry, but in that case you should be getting a PhD and not a master's, and also reading a different book than this one. The only time prestige can be useful is for the strong alumni networks these schools provide.
- **Your advisor [very important, but…]** – if the graduate program you're considering has a thesis or dissertation component, then you'll have an advisor from the school guiding you through it. Having an advisor with a similar work style and area of interest to you, along with not being an abusive person, will easily make the difference between succeeding in the program or not. Unfortunately it's very difficult to know before going to a school which advisor you'll pair with, nevermind what their personality is like. So while this is extremely important, you can only do so much to make decisions based on it. If however the program is entirely coursework based or has only a capstone project,

then an advisor won't matter much.

When coming up with your list of schools, try making a spreadsheet that lists how the different schools fare in each of these metrics. Once you have all the data, you may find it hard to objectively rank the schools: how can you really say if a school in a city you'd hate to live in but is well-connected to industry is better or worse than a school in a city you'd love that has no project work? We recommend you let go of the idea of finding the objective "best." Instead group the schools into "love," "like," and "okay" and then only apply to the loves and likes.

ONLINE GRADUATE PROGRAMS

More and more schools are offering online graduate programs, making it possible to learn everything you need to know without having to walk onto a college campus. The obvious benefit of this program is that having courses online is dramatically more convenient than having to spend hours each week going to a university. There also isn't nearly as much of a stigma from employers around online programs as there was at their inception, so you shouldn't worry if your degree will be viewed as legitimate. The downside is that it is *much* more difficult to stay engaged with the program and the material if you are doing everything online. It'll be harder to interact with your professors when you have questions, and it'll be easier to half pay attention and not do your homework. In a sense, that convenience of an online program can also be its downfall—you don't have as much incentive to stick with it. If you think that you have the ability to stay committed and focused to an online program, then they can be a great choice; just beware of the risks.

3.1.2 Getting into an academic program

To get into an academic program, you need to apply. If you're familiar with applying to graduate programs, the application process for data science master's degrees is similar to the rest. The first step is to write your application. Schools typically announce in the fall how to apply, including deadlines and materials needed. Graduate applications usually require:

- **A 1-2-page letter of intent** describing why you are a good fit for the program. For this letter, focus as much as possible on why you would make a good contribution to the program. Things like existing experience in some skills required by data science or examples of work you have done that is related are extremely useful. Try to avoid cliché statements like "how you have been interested in data science ever since you were a child." There are lots of resources available on writing good graduate school essays, and your undergraduate school may also have a department to help with this.
- **A transcript** from your undergraduate school. This is to show that you have the necessary prerequisites for the program. Your school's website should have instructions on how to get this, but keep in mind it usually costs money and takes a week or more. Don't leave it to the last minute!
- **GRE scores** that meet some minimum scores in verbal and math. The mathematics GRE should be relatively easy for anyone going into a data science program, but definitely spend time reviewing for it since you may have not seen the material in years. The verbal can be harder and may require substantial studying. The GREs

require you go to a special location to take the exam and can be a pain to schedule, so be proactive and try to get this done early. If English is not your native language, you'll probably need to get a minimum score on the TOEFL or IELTS.

- **Three letters of recommendation** on why you would be good for this graduate program. These can be from professors you've had or people like your boss if your job is tangentially related to data science. Ideally the writers should be able to talk about why you would be a good data scientist—so the people should have seen you perform well. Try to avoid college professors who can only say "this person got an A in my class" and employers who can't say much about your work in a technical environment. If you're an undergraduate student and you're reading this book, now may be a good time to get to know professors better by going to office hours, attending seminars, and joining academic clubs.

These different materials take time to pull together, and if you are applying to many schools at once it can end up being a full-time job. Most applications are due between December and February, and you then hear back around February or March. If you get accepted, you have until April to decide if you want in the program. When your acceptances come in, don't worry too much about which is exactly the best, just choose one that you think you'll be happy at!

3.1.3 Academic degree summary

Putting it all together, graduate programs in data science are a good fit for people who want an extensive education and can afford it. That could be a person who is coming from a field where they haven't done much programming or technical work, such as person who has been working in marketing. A graduate program would allow the person to learn all of the components of data science at a pace that would make the transition reasonable. Graduate programs are not good for people who already have a lot of the required skills. They are much too long and too expensive to be worth it. They also aren't taught by industry experts, so the little new knowledge they learn may not even be that relevant. You may need to get industry experience from internships during your graduate program to augment the degree itself.

If you're thinking you need extensive training before you could be a data scientist, then go for it and start looking for graduate schools you like. If you feel like this is going to be a lot of work and there's got to be an easier way, then consider the next few options.

Do I need a PhD?

Probably not.

PhDs are degrees that take many years to obtain and focus on training students to become professors. You have to spend years getting a new method or solution to a problem very slightly better than a previous one. You publish to academic journals and move the state-of-the-art research forward in an extremely specific area. But as the previous chapters have shown, little work that a data scientist does is like academic research. A data scientist cares much less

about finding an elegant solution that's state of the art and much more about quickly finding something that's good enough.
There are a fair number of data science job posts that require a PhD. But it's rarely the case that the skills gathered in a PhD are what's needed for the job—usually that's the company signaling that the position should be considered prestigious. The material you can learn from a Masters or undergraduate degree will suit you fine for the vast majority of data science jobs. There is a huge opportunity cost to getting a PhD. If it takes 7 years to graduate, you could have been instead working at a company for 7 years getting better a data science and making far more money.
So while you could go get a PhD and then be a data scientist, don't let anyone tell you you need one.

3.2 Going through a bootcamp

A bootcamp is an 8-15 week intensive course put on by companies like Metis and Galvanize. During the bootcamp you spend 8+ hours every day learning data science skills, listening to industry speakers, and working on projects. At the end of the course, you'll usually present a capstone project to a room full of people from companies looking to hire data scientists. Ideally, your presentation gets you an interview and then a job.

Bootcamps teach you an incredible amount of knowledge in a very short time. This means they can be great for people who have most of the skills needed for data science, but just need a bit more. For instance, consider someone who has been working as a neuroscientist and has done programming as part of her work. A data science bootcamp could teach her topics like logistic regressions and SQL databases. With her science background plus those basics, she should be ready to get a data science job. Sometimes the best part of a bootcamp isn't the knowledge itself, but the confidence you get from the program that you can actually do the work.

3.2.1 What you learn

A good bootcamp will have a syllabus that is highly optimized to teach you exactly what you need to know to get a data science job and little more. That goes beyond just technical skills and includes opportunities to work on projects and network with people. Here is more detail on what you should expect the program to cover.

SKILLS

Bootcamps are a great supplement to an existing education. For example, if you are someone who has been working as a software developer for years, a bootcamp can quickly fill in the details you need around the math and statistical techniques and how to think about data. By doing a bootcamp you'll be able to get a data science job quickly, without spending two years in a program like you would if you were to seek a master's degree. This might be especially attractive if you already have a master's degree in a non-data science field. The skills you typically get are:

- **Introductory statistics** – this includes methods for making predictions with data such

as linear and logistic regressions, as well as testing methods you could use on the job like *t*-tests. Due to the very limited time range, you won't get very deep into why these methods work, but you'll learn a lot on how to use them.

- **Machine learning methods** – you'll cover machine learning algorithms like random forests and support vector machines, as well as how to use them by splitting data into training and testing groups and using cross validation. You may learn algorithms for more specific cases like natural language processing or search engines. If none of those words made sense to you, that means you could be a good fit for a bootcamp!
- **Intermediate programming in R or Python** – you'll learn the basics around how data is stored in data frames, and how to manipulate it by summarizing, filtering, and plotting data. You'll learn how to do the statistical and machine learning methods within the chosen program. While you may learn R or Python, you probably won't learn both so you may have to learn the other after you finish the bootcamp if you need it for your first job.
- **Real-world use cases** – You'll not only learn the algorithms but also where people use them. Cases like using a logistic regression to predict when a customer will stop subscribing to a product, or how to use a clustering algorithm to segment customers for a marketing campaign. This knowledge is extremely useful for getting a job, and questions around use cases often show up in interviews.

PROJECTS

Bootcamps have a highly project-based curriculum. Instead of listening to lectures for 8 hours a day, most of your time will be spent working on projects that will best help you understand data science and get you started with your own data science portfolio (the subject of Chapter 4). That's a huge advantage over academia because your skills with be aligned with what you need to succeed in industry, since that's often similar to project-based work.

In a project, you'll first collect data. That could be through using a web API that a company has created to pull their data. It could also be through scraping websites to collect the information off them or using existing public data sets from places like government websites. You'll then load them into R or Python and write scripts to manipulate the data and run machine learning models on it. Then you'll use the results to create a presentation or report.

None of those steps in the project require a bootcamp (in fact Chapter 4 of this book exists just to help you do this yourself). That being said, having a project be part of a bootcamp means you'll have instructors guiding you and helping you if things go wrong. It's difficult to stay motivated if you are working alone and easy to get stuck if you don't have a person to call for help.

A NETWORK

Lots of people go on from bootcamps to successful careers at places like Google and Facebook. The bootcamps keep alumni networks that you can use to get your foot into the

door at those companies. The bootcamp may bring in data science speakers to talk to you during the program. There are also people from industry viewing your final presentations. These people can all serve as connections to help you get a job at one of their companies. Having points of entry to companies with data science positions can make all the difference when it comes to finding jobs, so this perk of bootcamps must be stressed.

In addition to meeting people during the course itself, you can also use tools like LinkedIn to contact alumni from your bootcamp. They may be able to help you find a job at their company, or at least point you in the direction of a company that would be a good fit.

For all of these connections, you'll have to be proactive. That means taking actions like going up and talking to speakers after they present and taking the initiative to send messages on social networks with people you haven't talked to before. This can be scary, especially if you aren't especially comfortable with social interaction with strangers, but it's necessary to get the value out of the bootcamp. Check out chapter 6 for tips on how to write an effective networking request.

3.2.2 Cost

One significant downside to the bootcamp compared to self-teaching is the cost: the tuition is generally around $15,000-$20,000. While you may be able to get scholarships to cover part of the tuition, you also have to consider the opportunity cost of not being able to work full-time (and likely even part-time) during the program. Moreover, you'll probably be on the job market for several months after your bootcamp. You won't be able to apply during the bootcamp because you'll be too busy and won't have learned the skills yet, and even a successful data science job application process can take multiple months from application to starting date. That can end up being an aggregate of 6-9 months of unemployment, in addition to the cost of the program. If you are able to teach yourself data science in your free time, or learn on the job, then you can keep working and not paying tuition, saving tens of thousands of dollars.

3.2.3 Choosing a program

Depending on where you live, there are likely only a few options for bootcamps. If you want to do an in-person bootcamp, even if you live in a large city there will probably be only a handful of programs. If you don't live in a large city and want to do a bootcamp, you may have to temporarily move to one. That can add to the cost of the program and make it more of an upheaval. Alternatively, there are online bootcamps for data science. Be careful, however: like with graduate programs, one of the benefits of in-person boot camps is that you'll have people around you to motivate you and keep you focused. If you do an online course you lose that benefit, which can make an online bootcamp a $20,000 version of the same courses you could get through free or cheap massive open online courses.

In selecting between the bootcamps in your area, consider checking out their classroom, talking to some of the instructors, and seeing where you feel the most comfortable. But

beware: with both academic degrees and bootcamps, there are lots of people looking to make a quick buck on people wanting to become data scientists. If you aren't careful, you can end up completing a program that doesn't help you get a job at all and leaves you with tens of thousands of dollars of debt. For bootcamps, it's extremely important that you talk to alumni. Do you see successful graduates on LinkedIn? If so, talk to them and see how they feel about their experience. If you can't find people on LinkedIn from the program that's a huge red flag.

3.2.4 Data science bootcamp summary

Bootcamps can be great programs for people wanting to switch careers and already have some of the basics of data science. They can also be useful for people just leaving school and wanting a few data science projects in their portfolio when on the job market. They are not designed to take you from "0 to 60" though; most of them have competitive admissions and you need to have a background in the fundamentals of programming and statistics to get in and then get the most out of it.

3.3 Getting data science work within your company

You may find yourself in a data science adjacent job. An unusual, but often very effective, method of learning data science is to start doing more and more data science work as part of your current job. Maybe you are a business person who adds a business spin to data science reports and could start also adding your own graphs. Or you work in finance making spreadsheets that you could move into R or Python.

For example, consider a hypothetical Alex, a person who has been working for several years in the market research department running surveys on customers and using a market research GUI (Graphical User Interface) to aggregate the survey results. Alex has a background in sociology and did a small amount of programming during their undergrad. Alex frequently works with the data science department–Alex passes them survey data and helps the data scientists understand it so they can use it in models. Over time, Alex starts to do a bit of work for the data science team. A bit of feature extraction in R here, a bit of creating visualizations there, and soon the data science team is relying more and more on Alex. During this time Alex is really improving their programming and data science skills, and after a year Alex fully joins the data science team, leaving market research behind.

Trying to do some data science in your current job is a great method because it is low risk and has built in motivation. You aren't trying to do an expensive bootcamp or degree that you have to quit your job for, you're just trying to add a little data science work where you can. And the fact that you are doing data science in your own job is motivating because the work you'll do is valuable to others. Over time you can do more data science work until eventually it's all you do, rather than trying to do an educational program and suddenly switch all at once.

Our former market researcher and now data scientist Alex had a number of things going for them in our example. Alex already had existing relationships with the data science

department to provide mentorship. Alex also understood the basics of programming and data visualization. Alex was self-motivated enough to learn data science techniques within their job. The data science department was able to provide Alex with small projects that Alex could tackle, and over time these grew to enable Alex becoming a data scientist. When trying to do more data science work in your company, look for places where you can find small data science projects and people to help you with them. Something as simple as creating a report, or automating an existing one, can teach you a lot of data science.

One important note when taking this path: *never become a burden for someone else*. That could be obvious, e.g. direct burdens might be repeatedly asking people to send you cleaned datasets. Or it could be less obvious things like constantly asking someone to review work you've done. You can also be an accidental burden by adding new tools to your team–if you're in finance and everyone uses Excel except for you who now uses R, you've just made managing your team more complicated. Even the task of asking a data scientist for work you can help with can be a burden. Just be thoughtful as you learn these skills that you aren't creating issues for other people.

TWO CONVERSATION PERSPECTIVES

What you say: "I am happy to help in any way I can, just let me know how! Thanks!"
What you think they hear: "I am a person who is eager to work for you, and you can hand me that exciting but simple project you've held onto for so long and I will do it for you!"
What they actually hear: "Hi! I want to be helpful but I have no idea what your needs are. I also don't know what my skills are relative to your workload, so good luck finding a task for me to do. Also, if you do somehow find a task that's perfect for me, you'll probably have to review it a bunch of times before it's good, all of this taking away from your already minimal free hours. Thanks!"

To make this path work there are a few key strategies to employ:

- **Be proactive** - the more you can do work before people ask for it, the more you'll become independent and less of a burden. For instance, the data science team may have a task, like labeling data or making a simple report, which is time intensive and uninteresting. You can offer to help with that work. Be careful just diving in entirely and doing it yourself–you may end up doing it in a way that instead of providing any value just provides the team the opportunity to redo your work. But if you can get it started and then get their input, it's possible you can save them a lot of time.
- **Pick off new skills one at a time** - don't just try to become a data scientist all at once. Find a single skill you want to learn through work and then learn it. For instance, you may want to learn how to make reports with R since the data science team does that all the time. By finding a small project to help the team with, you can pick up the skill and add it to your tool belt. From there you can learn a different data science skill and gradually learn enough that way.

- **Be clear with your intentions** - it'll be pretty obvious fairly quickly that you're trying to pick off extra work to learn to be a data scientist. By being proactive and letting the data science team know that you're interested in learning more, the team will be able to plan around having you help. It'll also make the team more understanding of your inexperience, since they were learning and new once too.
- **Avoid being pushy** - helping a person become a data scientist is an immense amount of work, and data science teams are often already overworked. If you find that the team doesn't have the time or bandwidth to help you, don't take it personally. While it's okay to occasionally check in if you think they've been out of touch, if you are too persistent with your requests the team will quickly become uncomfortable. They'll view you less as a potential resource and more as a nuisance.

When opportunities aren't there

You may find yourself in a situation where there are just *no opportunities for using data science in your current role. It's possible that the constraints of your job make it so that you couldn't possibly use R or Python or try implementing data science techniques. In these situations you may have to take more drastic measures. Quitting your job to do a bootcamp or graduate degree is risky but effective at getting you out of your current role and into data science. You could also try teaching yourself in your free time, but that has tons of drawbacks (see the section below). Another option is trying to find a new job in your field that would allow you to learn more in that position. But you can't guarantee that once you got to the new job you'd get the flexibility you were promised.*

None of these options are easy but unfortunately that's the reality of the situation. It's going to take work to get into a data science position, but it might well be worth it.

3.3.1 Learning on the job summary

Learning on the job can be an effective way to become a data scientist, provided you have a job where you can apply data science skills and people around who can mentor you. If those things align then this is a great route, but for many these things are not in place. If you do think this is a viable route for you, we highly recommend taking it–it isn't often that jobs allow for this so take the opportunity if you have it.

3.4 Teaching yourself

There are an enormous number of books covering data science [glances left, glances right…], as well as many online courses. These books and sites promise to teach you the basics of data science as well as the in-depth technical skills through a medium (and at a price point) that is practical. These courses and books, as well as all of the data science blogs, tutorials, and Stack Overflow answers, can provide enough of a grounding that people can teach themselves data science.

These self-driven learning materials are great for picking up individual skills. For instance, if you want to understand how to do deep learning, a book can be a great way to do it. Or if

you want to get the basics of R and Python, you can take an online course to get started with those.

Teaching yourself data science entirely through self-driven online courses and books is just like teaching yourself to play an instrument through YouTube videos or learning anything else without a teacher: its value is mostly a function of your perseverance. According to Malcolm Gladwell and all the people who repeatedly quote him, mastering a new skill takes 10,000 hours. It's really hard to put 10,000 hours into data science where the best Vine compilations are one tab over. It's also hard to know where to start; if you want to learn everything in data science who is to say which book you should read first [glances right, glances left…].

Learning on your own means you don't have a clear teacher or role model. By not having a teacher you can ask questions as you would in an academic program or bootcamp, you won't have an easy ability to understand when you're doing something wrong or know what to do next. This again places a hurdle to understanding the material. The best way to counteract this lack of a teacher is to find a community of people where you can ask questions. One great example is the TidyTuesday program started by Thomas Mock, in which every Tuesday aspiring and practicing data scientists use R to tackle a data science problem.

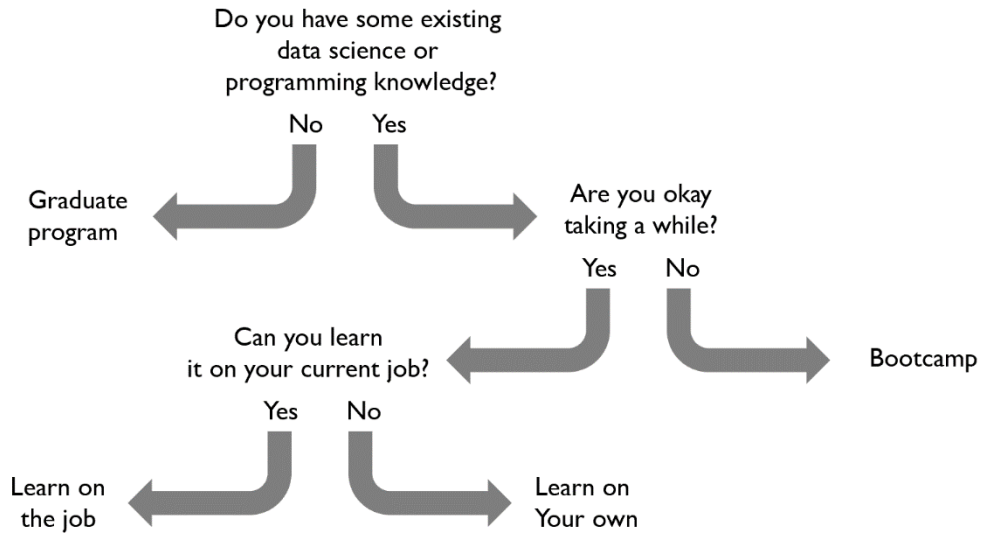
If you do decide to go down the self-driven route, it's important to have some constructive work you can do. While reading books and watching videos is great, you learn far more from doing your own data science work and learning from it. To put it a different way, reading books on bicycles can be educational, but you'll never learn to ride a bicycle without actually getting on one. So make sure to find a project that you want to do: taking a dataset and finding interesting results from it, creating a machine learning model and API, or using a neural network to generate text. In the next chapter we will go into far more detail on doing these sorts of projects. For other methods of learning data science, the projects can be about building a portfolio, but for self-driven learning, the projects are critical to learning.

3.4.1 Self-teaching summary

Learning on your own is hard; possible, but hard. You need to be able to figure out what order you want to learn things in, keep yourself motivated enough to learn the skills, and do it all without a clear mentor or teacher to help you. You'll also have more of a difficult time displaying your qualifications on a resume than if you used other methods. This is our least recommended way to become a data scientist, because of how many things can go wrong and how many people don't succeed in staying focused. If you want to pick up a single particular skill or technology it can be more feasible, but learning everything you need to be a data scientist is a hard route.

3.5 Making the choice

With those four very different avenues to data science, how do you pick? The process is different for everyone, but we propose answering these three questions:



Flow of deciding which data science education route to take

1. **Do you have some data science knowledge already?** Specifically, have you programmed in at least one programming language beyond light coursework? Do you know how to query data from a SQL database? Do you know what things like linear regressions are?
    a) If NO I'VE GOT A LOT TO LEARN, you'd probably be best suited for an academic program like a master's degree. This will teach you all of these topics over a long enough time period that they will really settle in.
    b) If YES I KNOW THIS STUFF, move on to question two.
2. **Are you comfortable with taking a year or more to gain the necessary data science skills, rather than incur the costs of being unemployed for six – nine months in order to get a data science job more quickly?** It's difficult to quickly learn new skills when you're solely focused on learning; it's even harder to do so while also working in a full-time job. Are you okay with the path taking longer so you can keep working full time?
    a) If NO I'VE GOTTA GO FAST, then take a bootcamp. In three months you'll know a ton of data science and be ready to embark on your search for a new job, which could take 3 – 6 months longer.
    b) If YES I WANT TO TAKE MY TIME, then move on to question three.
3. **Can you learn data science in your current job?** Can you do data science things within your current role? Maybe make an analysis, store some data in SQL, or try using R or Python? Is there a team that could mentor you or give you some small tasks?

a) If YES I CAN LEARN AT WORK, then try doing that and use your job as a springboard to data science.

b) If NO MY JOB HAS NO OPPORTUNITIES, then it's time to hit the books and online courses.

While these questions should provide you with a starting point, you don't have to make one final decision. You can start by reading books independently, and if you find you want to go quicker switch to a bootcamp. You can also get a master's degree in the evening while trying to do data science in your current job. There isn't one perfect answer, what matters is you find an approach that works for you, and if something isn't working you change it until it does.

Once you've chosen your route, it's time to do it! Enroll in that master's, join that bootcamp, or buy those books and start reading. For the purposes of this book we'll assume *~time passes~* and that you've succeeded in learning the fundamental skills you need to be a data scientist. In the next few chapters we'll take those skills and use them to create a data science portfolio and use that to help you get your first data science job.

3.6 Interview with Julia Silge, Data Scientist at Stack Overflow

Julia Silge is known for her popular blog posts on data science along with the tidytext package she and David Robinson developed, which is a cornerstone of NLP in R and has been downloaded more than 700,000 times. She and David also coauthored the book Text Mining with R: A Tidy Approach*.*

3.6.1 Before becoming a data scientist, you worked in academia; how have the skills you learned there helped you as a data scientist?

When I was doing research as an academic, some of my days were spent collecting real-world data. That experience taught me to think about the *process* the data is created by. In that case it was created by a physical process that I could touch. I could actually see things that contributed to why the data was messy, or why we didn't get a datapoint in a particular night. I see direct parallels to my work now at a tech company that deals with web data—there is some process that generates that data, and I think carefully about how we are recording that data and how that process can go well or wrong.

Another set of skills I learned before being a data scientist was communicating and teaching. I was a college professor for a number of years, and I also have had roles where I have dealt with customers. In those roles I practiced the skill of having a concept and trying to transfer that knowledge or understanding to another person. I strongly believe that's a part of most data scientists' role. If we train some model or do some statistical analysis, the value of it is small compared to when we are able to take that same model or analysis and explain what it means, how it works, or how to implement it in the broader context.

3.6.2 When deciding to become a data scientist, what did you use to pick up new

skills?

I certainly think academic programs, bootcamps, and online materials are all great options for different people in different situations. Given that I already had a PhD, I didn't want to go back to school and spend more money. I will admit that I applied to a couple of bootcamps and they took a pass on me! When I was deciding to make this career transition into data science, what I perceived was that I could do that job, but I needed to demonstrate to other people that I could. I also needed to update my machine learning knowledge and some of the techniques because when I was in grad school modern machine learning had not really entered astrophysics.

I went the route of online courses and embracing a lot of self-directed study. I joke around that I took every MOOC (massive open online course) that existed: it was a LOT. I took about six months where I was slightly under-employed in the job I had, and I just hit the MOOCs *really hard*. I had been out of school for a long time and I had been excited about the material. I hadn't been doing even data analysis for a while, so getting back into data analysis was really exciting!

3.6.3 Did you know going into data science what kind of work you wanted to be doing?

When I looked at my options ahead of time and saw what people were doing, like how people talked about "analyze" vs "build" data science, I absolutely saw myself as an analyze person. I saw myself less of an engineer and more of a scientist—a person who works to understand things and answer questions but not so much build things. And that has been where my career has started. I am the only data scientist at Stack Overflow and I am on a team with very talented, knowledgeable data engineers. But being the only data scientist, my job has shifted to be a bit of both data analysis and model building.

3.6.4 What would you recommend to people looking to get the skills to be a data scientist?

One thing that I would strongly emphasize is that you need to demonstrate that you can do this job. That can look different for different people. It's still a young enough field that people aren't sure what it means to be a data scientist and who can be one—it's still very undefined. There is still a lot of uncertainly on what this role means, and the positions are highly paid enough that the perceived risk to a company hiring wrong is very high, so companies are very risk adverse. Companies need to be sure that the candidate can do that job. Some ways I've seen people demonstrate that they can do the job is through open source contributions, speaking at local meetups on projects they've done, and developing a portfolio of projects on a blog or GitHub profile. For me, I took all the MOOCs and things I needed to learn and started a blog about all of these projects. What I pictured to myself was that these projects and blog posts would be something that we could talk about in a job interview.

3.7 Summary

- Four proven paths to learn the skills to be a data scientist are academic programs, bootcamps, picking the skills up in your current job, and teaching yourself on the side.
- Each of these methods has trade-offs in terms of the materials taught, the time they take, and the level of self-motivation required.
- To choose a path for yourself, take time to do introspection on what skills you already have, where your strengths lie, and what resources you have.

4

Building a Portfolio

This chapter covers:

- How to create a data science project
- Starting a blog
- Example projects

You've now finished a bootcamp, a degree program, a set of online courses, or a series of data projects in your current job. Congratulations—you're ready to get a data scientist job! Right?

Well, maybe. Part 2 of this book will be all about how to find, apply for, and get a data science position, and you can certainly start on this process now. But there's another step that can really help you be successful—building a portfolio. A portfolio is a set of data science projects that you can show to people so they can see what kind of data science work you can do.

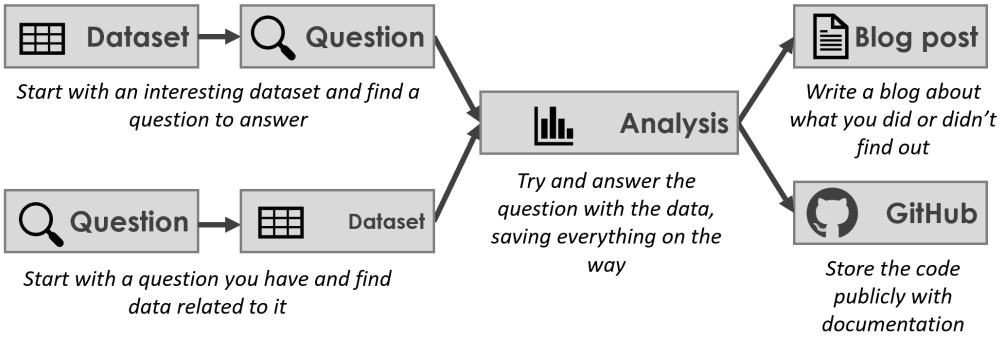
A strong portfolio has two main parts - GitHub repositories, or repos for short, and a blog. Your GitHub repo hosts the code for a project, and the blog shows off your communication skills and the non-code part of your data science work. Most people don't want to read through thousands of lines of code (your repo); they want a quick explanation of what you did and why it's important (your blog). And who knows, you might even get data scientists from around the world reading your blog, depending on the topic. As we'll cover in the second part of this chapter, you don't just have to blog about analyses you did or models you built; you could explain a statistical technique, write a tutorial for a text analysis method, or even share career advice, like how you picked your degree program.

This is not to say you need to have a blog or GitHub repos filled with projects to be a successful data scientist. In fact, the majority of data scientists don't, and people get jobs without a portfolio all the time. But it's a great way to help you stand out and to practice your data science skills and get better. Hopefully it's fun too!

This chapter will walk you through how to build a good portfolio. The first part is about doing a data science project and organizing it on GitHub. The second goes over best practices for how to start and share your blog, so that you get the most value out of the work you've done. Finally, we'll end by walking through two real projects we've done so you can see the process end-to-end.

4.1 Creating a project

A data science project starts with two things: a dataset that's interesting and a question to ask about it. For example, you could take government census data and ask, "how are the demographics across the country changing over time?" The combination of question and data is the kernel of the project, and with those two things you can start doing data science.



The flow of creating a data science project

4.1.1 Finding the data and asking a question

When thinking about what data you want to use, the most important thing is to find data that's interesting to you. Why do you want to use this data? Your choice of data is a way to show off your personality or the domain knowledge you have from your previous career or studies. For example, if you're in fashion, you can look at articles about fashion week and see how styles have changed in the past twenty years. If you're an enthusiastic runner, you can show how your runs change over time and maybe look to see if it's related to the weather.

Something you shouldn't do is use the titanic dataset, MNIST, or any other popular beginning datasets. It's not that these aren't good learning experiences; they can be, but you're probably not going to find anything novel. It won't surprise and intrigue employers or teach them more about you.

Sometimes you let a question lead you to your dataset. For example, you may be curious about how the gender distribution of college majors has changed over time and if that's related to the median earnings after graduation. You then would take to Google and try to find the best source of that data. But maybe you don't have a burning question you've just been

waiting to have data science skills to answer. In this case, you can start by browsing datasets and seeing if you can come up with any interesting questions. Here are a few suggestions of where you might start:

- **Kaggle:** Kaggle started as a website for doing data science competitions. Companies would post a dataset and a question and usually offer a prize for the best answer. Because the questions entailed machine learning projects where you're trying to predict something, like whether someone would default on a loan or how much a house will sell for, you can compare models based on their performance on a holdout test set and get a performance metric for each one. Kaggle also has discussion forums and "kernels" where people share their code, so you can learn how others approached the dataset. All of this means that Kaggle has thousands of datasets with accompanying questions and examples of how other people analyzed them.

  The biggest benefit of Kaggle is also its biggest drawback: by handing you a (generally cleaned) dataset and problem, they've done a lot of the work for you. You also have thousands of people tackling the same problem, so it's difficult to make a unique contribution. One way to use Kaggle is to take a dataset but pose a different question or do an exploratory analysis. But generally, we think Kaggle is best for learning by tackling a project and then seeing how you perform compared to others, thus learning from what their models did, rather than as a piece of your portfolio.

- **Datasets in the news**: Recently, many news companies have started making their data public. For example, FiveThirtyEight, a website focused on opinion poll analysis, politics, economics, and sports blogging, publishes data they use for their articles and even links to the raw data directly from the article website. While these data sets often require manual cleaning, the fact that they were in the news means there's probably an obvious question associated with them.
- **APIs**: APIs are Application Programming Interfaces - they are developer tools that allow you to access data directly from companies. You know how you can type in a URL and get to a website? APIs are like URLs but instead of a website you get data. Some examples of companies with helpful APIs are the New York Times and Yelp, which let you pull their articles or reviews respectively. Some APIs even have R or Python packages that specifically make it easier to work with them; for example, rtweet for R lets you quickly pull Twitter data, so you can find tweets with a specific hashtag, what the trending topics in Kyoto are, or what tweets Steven King is favoriting. Keep in mind that there are limitations and terms of service for how you can use them; for example, right now Yelp limits you to 5,000 calls a day, so you wouldn't be able to pull all reviews ever. APIs are great for providing extremely robust, organized data from many sources.
- **Government open data**: There is a lot of government data available online - you can use census data, employment data, the general social survey, and tons of local government data like New York City's 911 calls or traffic counts. Sometimes you can

download this data directly as a CSV; other times, you need to use an API. You can even submit freedom of information requests to government agencies to get data that isn't publicly listed. Government information is great because it is often detailed and on unusual subjects, like data on the registered pet names of every animal in Seattle. The downside of government information is that it is often not well-formatted, such as tables stored within PDFs.

- **Your own data**: There are many places where you can download data about yourself: social media websites and email services are two big ones, but if you use apps to keep track of your physical activity, reading list, budget, sleep, or anything else, you can usually download that data as well. Maybe you could build a chatbot based off your emails with your spouse. Or you could look at the most common words you use in your tweets and how that's changed over time. Perhaps you could track your caffeine intake and exercise for a month and see if it can predict how much and well you sleep. The advantage of using your own data is its guaranteed your project will be unique: no one else has ever looked at that data before!
- **Web scraping**: Web scraping is a way to extract data from websites that don't have an API, essentially by automating visiting webpages and copying the data. For example, you could create a program to search a movie website for a list of 100 actors, load their actor profiles, copy the list of movies they're in, and put that in a spreadsheet. You do have to be careful though - scraping a website can be against terms of use and you can be banned. You can check the robots.txt of a website to find out what they allow. You also want to be nice to websites - if you hit them too many times, you can bring down the site. But assuming terms allow it and you build in time between your hits, scraping can be a great way to get unique data.

What makes a side project "interesting"? Our recommendation is to pick a more exploratory analysis where any result will probably teach the reader something or demonstrate your skills. For example, creating an interactive map of 311 calls in Seattle, color-coded by category, clearly demonstrates your visualization skills and that you can write about the patterns that emerge. On the other hand, if you try to predict the stock market, you'll likely not be able to, and it's hard for an employer to assess your skills if you only have a negative outcome.

Another tip is to see what comes up when you google your question. If the first results are newspaper articles or blog posts answering exactly the question you were asking, you may want to rethink your approach. Sometimes you can expand upon someone else's analysis or bring in other data to add another layer to the analysis. But you may need to start the process over again.

4.1.2 Choosing a direction

Building a portfolio doesn't need to be a huge time commitment. Perfect is definitely the enemy of the good here. Something is better than nothing - employers are first and foremost

looking for evidence that you can code and communicate about data. You may be worried that people will look and laugh at your code or say, "Wow, we thought this person may be okay, but look at their terrible code!" It's very unlikely that this will happen. One reason is employers tailor their expectations to seniority level: you won't be expected to code like a computer science major if you're a beginning data scientist. Generally, the bigger worry is you can't code at all.

This is where it's also good to think about the areas of data science we covered in the first chapter. Do you want to specialize in visualization? Make an interactive graph using D3. Do you want to do natural language processing? Use text data. Machine learning? Predict something.

Use your project to force yourself to learn something new. Doing this kind of hands-on analysis will show you where the holes in your knowledge may be. When data you're really interested in is on the web, you'll learn web scraping. If you think a particular graph looks ugly, you'll learn tweaks to ggplot2. If you're self-studying, it's a nice way to solve the paralysis of not knowing what to learn next.

A common problem with self-motivated projects is over-scoping. This is where you just want to do everything, or you keep adding more stuff as you go. You can always keep improving/editing/supplementing, but that means you never finish. One strategy is to think like Hollywood: sequels. You should set yourself a question and answer it, but if you think you might want to revisit it later, you can end your research with a question or topic for further investigation or even a "TO BE CONTINUED…?" if you must.

Another problem is not being able to pivot. Sometimes the data you wanted isn't available. Or there's not enough of it. Or you're not able to clean it. This is frustrating and it can be easy to just give up at this point. But it's worth trying to figure out how you can salvage it. Do you have enough already to write a blog post tutorial, maybe on how you collected the data? Employers look for people who learn from their mistakes and aren't afraid of admitting them. Just showing what went wrong so others might avoid the same fate is still valuable.

4.1.3 Filling out a GitHub README

Maybe you're in a bootcamp or a degree program where you're already doing your own projects. You've even committed your code to GitHub. Is that enough?

Nope! A minimal requirement for a useful GitHub repository is filling out the README. There are couple of questions to answer here:

- **What is the project?** For example, what data does it use? What question is it answering? What was the output - a model, a machine learning system, a dashboard, or a report?
- **How is the repository organized?** This implies, of course, that the repo is in fact organized in some manner! There are lots of different systems, but a basic one is just splitting up your script into different parts - getting (if relevant) your data, cleaning it, exploring it, and the final analysis. This way, people know where to go depending on

what they're interested in. It also suggests that you will keep your work organized when you go to a company. A company doesn't want to risk you working there and then, when it's time to hand-off a project, you give someone an uncommented, five-thousand-line script that may be impossible for them to figure out and use. Good project management also helps the you of the future - if you want to reuse part of the code later, you'll know where to go.

But while doing a project and making it publicly available on a documented GitHub repo is good, it's very hard to look at code and understand why it's important. After you do a project, the next step is to write a blog, which will let people know why what you did was cool and interesting. No one cares about pet_name_analysis.R, but everyone cares about "I used R to find the silliest pet names!"

4.2 Starting a Blog

Blogs allow you to show off your thinking and projects, but they can also offer a non-technical view of your work. We know, we know, you just learned all this great technical stuff! You want to show it off! But being a data scientist almost always entails communicating your results to a lay audience, and a blog will give you experience translating your data science process into "business" language.

4.2.1 Potential topics

Let's say you create a blog. Are people really going to be interested in your projects? You don't even have a "data scientist" title yet; how can you teach anyone anything?

Something good to remember here is that *you are best positioned to teach the people a few steps behind you*. Right after you've learned a concept, like using continuous integration for your package or making a TensorFlow model, you still understand the misconceptions and frustrations you've had. Years later, it's hard to put yourself in that beginner's mindset. Have you ever had a teacher that was clearly very smart and yet couldn't communicate concepts at all? You didn't doubt they knew the topic, but they couldn't break it down for you and seemed frustrated that you didn't just "get it" right away.

Try thinking of your audience as the you of six months ago - what have you learned since then? What resources do you wish had been available? This is also great for celebrating your progress. With so much to learn in data science, it's easy to feel like you're never enough; pausing to see what you've accomplished is nice.

You can group data science blog posts into a couple of types:

- **Code-heavy tutorials:** These teach your readers how to do things like web scraping or deep learning in Python. Your readers will generally be other aspiring or practicing data scientists. While we have called them "code-heavy," you'll usually still want there to be as many lines of text as code, if not more. Code is generally not self-explanatory; you need to walk a reader through what each part does, why you'd want to do it, and

what the results are.

- **Theory-heavy tutorials:** These teach your readers a statistical or mathematical concept, like what Empirical Bayes is or how Principal Component Analysis works. They may have some equations or simulations. Just like code-heavy tutorials, your audience will usually be other data scientists, but you should write so that anyone with some mathematics background can follow along. These are an especially good way to demonstrate your communication skills; there's a stereotype that many technical people, especially if they have a PhD, can't explain concepts well.
- **A fun project you did**: As we hopefully convinced you in the side projects section, you don't need to only work on groundbreaking medical image recognition. You can also find out which of the Twilight movies used only words found in Shakespeare's The Tempest. For example, Julia Silge, who we interviewed in chapter 3, used neural networks to generate text that sounds like Jane Austen. These can be focused more on the results or the process, depending on what the most interesting part of your project was.
- **Writing up your experience:** You don't just have to blog about tutorials or your data science projects. You can talk about your experience at a data science meetup or conference - what talks you found interesting, advice for people going to their first one, or some resources the speakers shared. This can be helpful to people considering attending the same one the following year or who can't attend conferences for logistical or financial reasons, and again provides an insight for employers into how you think and communicate.

4.2.2 Logistics

But where should you put your interesting writing? For creating a blog, you have two main options:

- **Making your own website**. If you work in R, we suggest using the package blogdown, which lets you create a website for a blog using R code (wild, right?). If you use Python, Hugo and Jekyll are two options, which both let you create static blog websites. Both come with a bunch of themes that other people have built and let you write blog posts in markdown. We suggest not worrying that much about your theme and style and just picking one you like—nothing is worse than not writing blog posts because you got too distracted by constantly changing the blog's appearance. Simple is probably best; it can be a pain to change your theme, so it's better not to pick one you may get sick of in six months.
- **Using medium or another blogging platform**. Medium is a free, online publishing platform. The company doesn't generally write content themselves; instead, they host it for millions of authors. Medium or other sites like it are good if you just want to start quickly because you don't have to worry about hosting or starting a website – all you do is hit "new post," start writing, and publish. You can also get more traffic when

people search the blogging site for terms like "data science" or "python." But one concern is you are at the company's mercy - if they change their business model and put everything behind a paywall, you can't do anything to keep your blog posts free. You also don't get to create a real biography section or the ability to add other content, like a page with links to talks you've given.

One common question people have when starting blogging is how often they need to post and how long those posts should be. This is definitely a personal choice. We've seen people who have more "micro" blogs where they publish short posts multiple times a week. Other people go months between posts and publish longer form articles. There are some limitations here: you do want to make sure your posts don't start to resemble Ulysses. If your post is very long, you can split into multiple parts. But you want to show you can communicate concisely, as that's one of the core data science skills. Executives and even your manager probably don't want or need to hear all the false starts you had or the twenty different things you tried. While you may choose to include a brief summary of your false starts, you need to quickly get to the point and your final path, and only include them if you ultimately found something. The exception is if the method you used is going to be surprising - for example, if you didn't use the most popular library for something because you found it didn't work.

What if you're worried that no one will read it and all your work will be for nothing? Well, one reason to have a blog anyway is that it helps your job applications. You can put links to your blog posts on your resume when you reference data science projects, and even potentially show them to people in interviews, especially if it's a nice interactive visualization or dashboard. It's not important to have hundreds or thousands of readers. While it can be nice if you get claps on Medium or you're featured in a data science company's newsletter, it's more important to have audience that will read, value, and engage with the material than to have high metrics.

That's not to say there's nothing you can do to build a readership. You should also advertise for yourself; although it's a cliché, having a #brand is useful for building a network in the long-term. Even if something seems simple, it's probably new to a bunch of practicing data scientists just because the field is so big. People at the companies you want to work for may even read your stuff! Twitter is a good place to start; you can share when you release a post and use the appropriate hashtags to get some wider readership.

But your blog is valuable even if no one (besides your partner and pet) reads it. Writing a blog post is good practice; it forces you to structure your thoughts. Just like teaching in person, it also helps you realize when you don't know something as well as you thought you did.

4.3 Example Projects

Let's walk through two example projects, from the initial idea through the analysis to a final public artifact. We'll use real ones that the authors of this book did: creating a web application

for data science freelancers to find the best fit jobs, and learning neural networks by training one on a dataset of banned license plates

4.3.1 Data science freelancers – a project by Emily Robinson

THE QUESTION

When I was an aspiring data scientist, I became interested in one way some data scientists make extra money: freelancing. Freelancing is doing projects for someone you're not employed by, whether that's another person or for a large company. These projects ranging from a few hours to months of full-time work. You can find many freelancing jobs posted on freelancing websites like UpWork, but since data science is a very broad field, jobs in that category could be anything from web development to an analysis in Excel to natural language processing on terabytes of data. I decided to see if I could help freelancers wade through thousands of jobs to find ones that are the best fit for them.

THE ANALYSIS

To gather the data, I used the UpWork's API to pull currently available jobs and the profiles of everyone in the Data Science and Analytics category. I ended up with 93,000 freelancers and 3,000 jobs. Although the API made it relatively easy to access the data (as I didn't have to do web-scraping), I still had to make functions to do hundreds of API calls and handle when it failed and transform the data so I could use it. But the advantage of this process was that since the data wasn't available easily, there weren't hundreds of other people working on the same one like there would be if I used data from a Kaggle competition.

After I got the data in good shape, I did some exploratory analysis. I looked at how education levels and country affected how much freelancers earned. I also made a correlation graph of the skills freelancers listed, which showed the different types of freelancers: web developers (PHP, jQuery, HTML, and CSS), finance and accounting (financial accounting, bookkeeping, and financial analysis), and data gathering (data-entry, lead generation, data mining, web scraping), along with the "traditional" data science skillset (Python, machine learning, statistics, and data analysis).

Finally, I created a similarity score between profile text and the job text and combined that with the overlap in skills (both freelancers and jobs listed skills) to create a matching score for a freelancer and a job.

THE FINAL PRODUCT

In this case, I didn't end up writing a blog post. Instead, I made an interactive web application where someone could enter their profile text, skills, and requirements for jobs (e.g. a minimum feedback score for the job poster, how long the job would take) and the available jobs would be filtered to meet those requirements and sorted by how well they fit you.

I didn't let perfect be the enemy of the good here; there are plenty of ways I could have made the project better. I only pulled the jobs once and, since I did this three years ago, while the application still works, none of those jobs are available anymore. To make it a valuable application in the long-term, I'd need to pull jobs nightly and update the listings. I also could have made a more sophisticated matching algorithm, sped up the initial loading time of the app, and make the appearance fancier. But despite these limitations, the project still met a few important goals. It showed I could take a project and allow people to interact with it, rather than be limited to static analyses that lived on my laptop. It had a real-world use-case: helping freelancers find jobs. It took me through the full data science project cycle, from gathering the data, cleaning it, running exploratory analyses, and producing a final output.

4.3.2 Training a neural network on offensive license plates – a project by Jacqueline Nolis

THE QUESTION

As I grew as a data scientist I was continuously getting frustrated when I saw hilarious blog posts online of people training neural networks to generate things like new band names, new Pokémon, and weird cooking recipes—I thought they were great but I didn't know how to make them myself! One day I remembered that I had heard of a dataset of all of the custom license plates that were rejected by the state of Arizona for being too offensive. If I could get that dataset it would be perfect for finally learning how to make neural networks.

Sample output of the offensive license plate generator neural network

THE ANALYSIS

After submitting a public records request to the Arizona Department of Transportation I got a list of thousands of offensive license plates. I didn't know anything about neural networks so after receiving the data I started scouring the internet for blog posts describing how to make one. As a primarily R user I was happy to find the Keras package by RStudio for making neural networks in R.

I loaded the data into R and then checked out the RStudio Keras package example for generating text with neural networks. I modified their code to work with license plate data—the RStudio example was for generating sequences of long text, but I wanted to train on 7-character license plates. This meant creating multiple training datapoints for my model from each license plate (one datapoint to predict each character in the license plate).

I then trained the neural network model, although at first it wasn't working at all. After putting the project down for a month, I came back and realized my data wasn't being processed quite right—once I fixed this, the results that the neural network generated were fantastic. Ultimately, even though I didn't change the RStudio example much, by the end I felt much more confident in creating and using neural networks.

THE WRITE-UP

I wrote a blog post about the project which walks through how I got the data, the act of processing it to be ready for the neural network, and how I modified the RStudio example code to work for me. The blog post was very much a "I'm new at neural networks and here is what I learned" style of post—I didn't pretend that I already knew how all of this worked. As part of the blog post I made an image that took the text output of my neural model and made it look like Arizona license plates. I also put the code on GitHub.

Since writing that blog post and making my code available, numerous other people have modified to make their own funny neural networks. What I learned from this goofy project eventually helped me on important consulting engagements to make high-impact machine learning models. Just because the original work isn't serious doesn't mean there isn't value in it!

4.4 Interview with David Robinson, Data Insights Engineering Manager at Flatiron Health

David Robinson is the co-author with Julia Silge the tidytext package in R and the O'Reilly book Text Mining with R. He's also the author of the e-book Introduction to Empirical Bayes and the R packages broom and fuzzyjoin. He holds a PhD in Quantitative and Computational Biology from Princeton University. David writes about statistics, data analysis, education, and programming in R on his popular blog, varianceexplained.org.

4.4.1 How did you start blogging?

I first started blogging when I was applying for jobs near the end of my PhD, as I realized that I didn't have a lot out on the internet that showed my skills in programming or statistics. When I launched my blog, I remember having the distinct fear that once I wrote the couple of posts I had ready, I would run out of ideas. But I was surprised to find that I kept coming up with new things I wanted to write about: datasets I wanted to analyze, opinions I wanted to share, and methods I wanted to teach. I've been blogging moderately consistently for four years since then.

4.4.2 Are there any specific opportunities you have gotten from public work?

I did get my first job from something I wrote publicly online. Stack Overflow approached me based on an answer I'd written on Stack Overflow's statistics site. I'd written that answer years ago, but some engineers there found it and were impressed by it. That experience really led me to have a strong belief in producing public artifacts, because sometimes benefits will show up months or years down the line and lead to opportunities I never would have expected.

4.4.3 Are there people you think would especially benefit from doing public work?

People whose resumes might not show their data science skills and don't have a typical background, like having a PhD or experience as a data analyst, would particularly benefit from public work. When I'm evaluating a candidate, if they don't have those kinds of credentials, it's hard to say if they'll be able to do the job. But my favorite way to evaluate a candidate is to read an analysis they've done online. If I can look at some graphs someone created, how they explained the story, and how they dug into the data, I can start to understand whether they're a good fit for the role.

4.4.4 How has your view on the value of public work changed over time?

The way I used to view projects is that you made steady progress as you kept working on something. In graduate school, an idea wasn't very worthwhile, but then it became some code, a draft, a finished draft, and finally a published paper. I thought that along the way my work was getting slowly more valuable.

Since then I realized I was thinking about it completely wrong. Anything that is still on your computer, however complete it is, is worthless. If it's not out there in the world, it's been wasted so far, and anything that's out in the world is much more valuable. What made me realize this is a few papers I developed in graduate school that I never published. I put a lot of work into them, but I kept feeling they weren't quite ready. Years later, I've forgotten what's in them, I can't find them, and they haven't added anything to the world. If along the way I'd written a couple of blog posts, done a couple of tweets, and maybe made a really simple open source package, all of those would have added value along the way.

4.4.5 How do you come up with ideas for your data analysis posts?

I've built up a habit that every time I see a dataset, I'll download it and take a quick look at it, running a few lines of code to get a sense of the data. This helps you build up a little of data science taste –working on enough projects that you get a feel for what pieces of data are going to yield an interesting bit of writing and which might be worth giving up on.

My advice is whenever you see the opportunity to analyze data, even if it's not in your current job or you think it might not be interesting to you, take a quick look and see what you can find in just a few minutes. Pick a dataset, decide on a set amount of time, do all the analyses that you can, and then publish it. It might not be a fully polished post, and you might

not find everything you're hoping to find and answer all the questions you wanted to answer. But by setting a goal of one dataset becoming one post you can start getting into this habit.

4.4.6 What's your final piece of advice for aspiring and junior data scientists?

Don't get stressed about keeping up with the cutting edge of the field. It's tempting when you start working in data science and machine learning to think you should start working with deep learning or other advanced methods. But remember that those methods were developed to solve some of the most difficult problems in the field. Those aren't necessarily the problems that you're going to face as a data scientist, especially early in your career. You should start by getting very comfortable transforming and visualizing data, programming with a wide variety of packages, and using statistical techniques like hypothesis tests, classification, and regression. It's worth understanding these concepts and getting good at applying them before you start worrying about concepts at the cutting edge.

4.5 Summary

- Having a portfolio of data science projects shared on a GitHub and a blog can help you get a job.
- There are many places you can find good datasets for a side project; the most important thing is it's something interesting to you and a little bit unusual.
- You don't just have to blog about your side projects; you can also share tutorials or your personal experience with a bootcamp, conference, or online course.

Part 2: Finding your data science job

Now that you're equipped to get a data science job, it's time to do it. This part of the book covers everything you need to know to conduct a successful job search, starting with finding open positions and ending with negotiating and accepting a job offer. The job hunt process in data science has some unique quirks because of the nature of the field. For example, we'll prepare you for parsing the many different job postings that mean "data scientist" and what companies are looking for in a take-home case study. While this part is especially useful if you haven't had a data science job before, the material can still be useful as a refresher for junior and senior data scientists.

Chapter 5 covers searching for data science positions and how to handle the dizzying array of job postings. Chapter 6 teaches you how to create a strong data science resume and cover letter, providing examples to base your material on and the principles behind them. Chapter 7 is all about what to expect from and how to prepare for the data science job interview, from the initial phone screen to the final on-site. Chapter 8 walks through what to say when you receive an offer from a company, including how to decide to whether to accept it and why and how to negotiate.

5

The Search: Identifying the Right Job for You

This chapter covers

- Finding open jobs that might be a good fit
- Decoding job descriptions to understand what the role is really like
- Picking which jobs to apply for

You've got the skills, you've got the portfolio – all you're missing is the data science job! The job search process does take some time – even successful job applications usually take at least a month from applying to getting an offer, and more commonly several months. But by following the best practices we lay out, we hope to make the process as painless as possible.

In this chapter, we'll focus on how to look for data science jobs. You'll first learn all the places where you can find jobs, making sure you won't unknowingly narrow your options. We'll cover how to decode these descriptions to find out what skills you actually need (spoiler: it's not all of them) and what the job might be like. Finally, you'll learn how to choose which ones are best suited for you, using the knowledge you've gained about data science skills and company archetypes from the first chapters.

5.1 Finding jobs

Before worrying about crafting the "perfect" resume and cover letter, you need to know where to send them! Job boards like LinkedIn, Indeed, and Glassdoor are a good place to start your search. It's worth looking at more than just one website, since not all companies will post on each one. If you're part of an underrepresented group in tech, you should also look for job sites targeted specifically at you, such as POCIT and Tech Ladies, for people of color and

women in technology respectively. The type of job you're applying to might also influence where you look - there are job boards for specific types of companies like start-ups (AngelList) and technology (Dice).

Make sure to browse widely. As discussed in the first chapter, data science jobs go by many names besides data scientist. Different companies use different names for similar roles, and some are even are changing what their titles mean, so all the people who were data analysts one year might be data scientists the next, with no change in responsibility!

Some examples of titles you might encounter include:

- **Data analyst**: This is more often a junior position and can be a great way to start in the field if you don't have a STEM degree and haven't done any data analysis for a company before. As we'll discuss later in this chapter, you do want to be extra careful with data analyst positions to make sure that the role will involve programming and statistics or machine learning.
- **Quantitative, product, research or other type of analyst**: These roles have even more diversity than data analyst in terms of your responsibilities. You may be doing exactly the same type of work as "data scientists" at other companies, or you may be spending your days with legacy Excel spreadsheets.
- **Machine learning engineer**: As implied by the title, these focus on the machine learning part of data science. They'll usually ask for a strong engineering background – if you have a degree in computer science or have been working as a software engineer, this could be a great role.
- **Research scientist**: These positions will often require a PhD, though there might be some negotiation room if you have a master's in computer science, statistics, or a closely related field.

When you're first starting your search, try searching for simply "data" on one of these job boards and spending an hour reading through job posts. This will give you a better idea of what industries are represented in your area and what types of positions are open. You'll pick up on patterns that will let you skim through new listings more quickly. Finding jobs that are a good match for you, rather than all the jobs that are available, will narrow the field down to a manageable number. Don't worry too much about the title – use the description to evaluate fit.

Data analyst
entry level
Analyze data & create reports

Product analyst
job varies
Focuses on one part of the company

ML engineer
software focused
Build ML models to power the business

Research scientist
theoretical
Research focused job, requires advanced degree

Some job titles which don't include "data science" that you may find when searching

Be extremely cautious about thinking of job-hunting as a numbers game. If you're looking in a big tech city like New York or San Francisco or in multiple cities, you'll find hundreds of jobs listed. Checking job boards can quickly become an obsession. It's an easy way to feel productive – "I read through 70 job descriptions today!" And just like Twitter and Facebook, checking constantly for updates can be addictive. But checking more than every three to five days doesn't generally add value. While checking only once a month could mean you miss out on a good opportunity, there's no company that has filled a position (that was actually open) within two days of posting on a job board.

If you have specific companies you're interested in, check out their careers page. Just as you should search for multiple job titles, check different departments - some companies may put data science under finance, engineering, or other departments, so if you hadn't checked there, you wouldn't have found them.

> **NEW GRADUATES** If you're about to or just graduated college, your most relevant skill is your education. Your data science portfolio will be helpful here too. When you're searching for jobs, look for positions specifically titled "New Grad," "Junior," "Associate," and "Entry-level." Also look at your career center for help and go to any job fairs that happen on campus. Internships are relevant – less for what work skills you learned and more that it shows you can come to an office each day, be professional and productive.

5.1.1 Decoding descriptions

Once you start reading job descriptions, it can seem like data science job postings fall into one of two categories. One is a business analyst position where you'll use business intelligence tools like Excel and Tableau, with maybe a little SQL, but generally you won't code. If you want to grow your coding skills, machine learning toolbox, or statistics and data engineering knowledge, these are not a good fit.

A common listing at the other extreme describes a unicorn - a PhD in Computer Science who's also worked for 5+ years as a data scientist and is an expert in cutting-edge statistics, deep learning, and communicating with business partners - and lists a huge range of responsibilities, from doing production-level machine learning to creating dashboards to running A/B tests. These types of job descriptions usually mean the company doesn't know what they're looking for, and they expect a data scientist to come and solve all their problems without any support.

Don't worry though – we promise there are more than these two types. The better way of thinking about these jobs is in terms of experience. Are they looking for someone to build a department of their own with no data pipeline infrastructure in place, or are they looking for a fifth member of a currently productive data science team where they hope that person contributes immediately, but isn't expected to be an expert in data manipulation, business communication and software development all at once? To do this, you need to take a job description and figure out what they're actually looking for. Imagine if you were looking at cat adoption listing and the cat Honeydew Melon is stated as "liking to ask about your day." You'd want to realize that actually means she will constantly meow for attention, which could be bad for your home. In job descriptions, some famous examples include "work hard and play hard" meaning you'll need to work long hours and be expected to attend informal company events (like going to bars) or "self-starter and independent" meaning you won't get a lot of support. By knowing how to read between the lines, you can make sure you'll be applying for the right jobs.

The first thing to keep in mind is that job descriptions are generally wish lists with some flexibility. If you meet 60% of the requirements (e.g. you're a year short of their required work experience or haven't worked with one component of their tech stack), but are otherwise a good fit, you should still apply. Definitely don't worry too much about the "plusses" or "nice to haves." Additionally, years of work experience requirements are just a proxy for having the necessary skills - if you coded in grad school, that could count. That being said, applying to a post for a senior data scientist that requires 5 years of work experience as a Data Scientist, proficiency in Spark and Hadoop, and experience deploying machine learning models into production is probably not the best use of your time if you're an aspiring data scientist – they're looking for a different level of experience and qualifications.

> **DEGREE REQUIREMENTS** Many data scientist jobs list a degree in a "quantitative discipline," meaning fields like Statistics, Engineering, Computer Science, or Economics, as a requirement. If you don't have one, can you still apply for them? Generally, yes. We'll discuss this more in the next chapter, but if you took classes in those areas (including in a bootcamp or online), you can emphasize that. If you followed the advice of the last chapter and built a portfolio and wrote blog posts, you can show those to employers as evidence you can do the work.

One complication in data science postings is that there are different words for the same things. Machine learning and statistics are infamous for this. One company may ask for

experience in regression or classification, another in supervised learning, but these are overall the same thing. The same goes for the terms A/B testing, online experimentation, and randomized control trials. If there's a term you're not familiar with, google it – you may find you've done it under a different name! If you haven't worked with a specific technology they're referencing, see if you've done something similar. For example, if they cite AWS (Amazon Web Services), and you've worked with Azure or Google Cloud, you have the skill of working with cloud-based technology.

The other benefit of knowing how to decode a job description is the ability to detect red flags. No company is going to straight up say that they are bad to work for. The earlier you recognize a likely bad work situation, the better, so you'll want to start looking for any warning signs in the job description.

5.1.2 Watching for Red Flags

Finding a job is a two-way street. It can feel during this process that companies have all the power and you need to prove you're deserving. But you, yes *you*, can also be selective about where you work. Ending up in a toxic workplace or a mind-numbingly boring job is a really hard situation. While you won't always be able to tell if this will be the case just from a job description, there are a few warning signs to watch out for.

The first is if there's no description of the company or job itself, just a list of requirements. Those organizations have forgotten it's a two-sided process and aren't thinking about you. It can also mean they're buying into the data science hype and just want to have a team of data scientists, without setting anything up so they can work productively.

A second warning sign is the aforementioned "unicorn" description. While that was an extreme example, you should be careful of any job description that describes two or three of the job types (decision science, analytics, and machine learning) as primary responsibilities. While it's normal to be expected to have base competency in each, no person is going to be able to fill all those roles at an expert level. Even if someone could, they wouldn't have time to do it all.

Finally, look for mismatches between the requirements and the description of the position. Are they asking for experience in deep learning, but the job functions are making dashboards, communicating with stakeholders, and running experiments? If so, they may be just wanting someone who can use the hottest tool or who's a "prestigious" data scientist, with a Stanford PhD in AI, when actually they can't use that specialized knowledge.

5.1.3 Setting your expectations

While you should have standards for a potential job, you don't want to demand perfection. Aspiring data scientists sometimes see their path broken down this way: "Step 1-98: Learn Python, R, deep learning, Bayesian statistics, cloud computing, A/B testing, D3. Step 99: Get a data science job. Step 100: Profit." While this is an exaggeration, part of the data science hype is the idealization of what it's like to work in the field. After all, data scientist is "the best

job in America" (according to Glassdoor), with a six-figure paycheck and high job satisfaction. You might imagine getting to spend every day on the most interesting problems in the field with the smartest colleagues. The data you need will always be accessible, cleaned, and any issues you face will be solved immediately by a team of engineers. Your job will be exactly as it was described, and you'll never have to do the parts of data science that interest you less.

Unfortunately, this is a fantasy. Just like we hope the first chapters convinced you that you don't need to know everything before getting into the field, companies aren't going to be perfect unicorns either. There's a reason this book doesn't end with you getting a data science job. Although it's a great accomplishment and you should be proud, data science is a field where you'll always be learning. Models will fail, workplace politics will scrap the work you've been doing for the past month, or you'll spend weeks working with engineers and product managers to collect the data you need.

It's especially easy to idealize companies that are well-known, either generally or for data science. Maybe you went to a talk and one of their employees blew you away. Maybe you've been following their blog for months and know they're on the cutting-edge of the field. Maybe you read an article about how they have nap pods, gourmet meals, and lots of friendly office dogs. But whatever has attracted you has likely interested other aspiring data scientists as well; most of these companies get hundreds of applications for an open position and can set the bar higher than even what is needed to do the job. In any case, the work you read about might be in a totally different division and this position may be uninteresting.

Even with realistic expectations, you'll likely not end up in your dream job for your first data science role. It's easier to transition within your field or bring data science into your current role; even if you're looking to eventually leave your domain, you may need to start out by moving to a position where you can leverage your other skills. That doesn't mean you shouldn't have certain requirements and preferences, but it does mean you'll want to have some flexibility. It's very normal to switch jobs in tech even after just a year or two, so you're not signing yourself up for the next 15 years. But you can't know exactly what you want before you're even in the field, and you'll learn even from bad jobs, so don't stress too much.

But you're not limited to only looking at job boards. As we'll discuss in the next chapter, submitting your application "cold" online often has a very low response rate. According to a survey by Kaggle in 2017, the two most common ways people already employed as data scientists look for and get jobs is through recruiters and friends, family, and colleagues (Kaggle.com/surveys/2017). A great way to build that network is by going to meetups.

5.1.4 Attending Meetups

Meetups are generally in-person meetings on weekday evenings. There usually will be a speaker, panel, or series of speakers presenting on a topic relevant to the meetup. Meetups should be free or only have a nominal fee which sometimes goes towards food. Some meetups may only have 20 people, while some may fill a room with 300. Some have meetings every month while others meet only a few times a year. Some encourage members to stay at the space afterwards to talk or meet at a nearby bar, where others are more focused just on the

talk itself. Some have very specific focuses, such as advanced natural language processing in Python, while others may offer an introduction to time series one month and advanced deep learnings models the next. It's worth trying a few and seeing what you enjoy most – while the topic is important, you want to find somewhere you feel welcome and enjoy talking with the other attendees. Almost all meetups have accounts on meetup.com, so you can search for "data science" or "analytics" in your area to find relevant meetups.

Many data science meetups have time at the beginning where people can announce if they're hiring. Go up and talk to these people; it's part of their job, and even if their current openings aren't a good fit they maybe be able to give you some good advice or suggest other places to look.

You may also meet another attendee who works in the company or sub-industry you're interested in. You can ask if they have time for an informational interview so you can learn more about their field. An informational interview isn't (or rather shouldn't be) a passive-aggressive way of looking for a referral – instead, it's a great way to get a look inside a company and get advice from someone who's in the field. While we'll talk in the next chapter about the advantages of being referred for a job, we don't recommend asking people you've just met to refer you. That's a strong ask for someone they don't know, and no one likes feeling like they're being used. If they tell you about an opening at their company and say they can refer you, that's a great bonus, but you'll gain a lot even if they don't.

Attending meetups is also great for other reasons. One, it allows you to find like-minded people who are local. If you moved to a new city or just graduated college, you might feel like a stranger in your town. This is a great chance to develop your career and also build your social circle. You can also take advantage of meetups to network or build some contacts who might be able to help you with anything from specific data science questions to offering job-hunting recommendations or general mentorship. Second, while some meetups post recordings of their talks online, others don't and thus attending in person is the only way to hear that talk.

Unfortunately, there can be a few drawbacks to meetups. It can be daunting to join a meetup that's only a few people who are all experienced or know each other. Imposter syndrome can definitely creep in, but you should fight through that as there are few more welcoming spaces than a good meetup. Finally, while meetups offer a great chance to see the local data science scene, they can be insular or lack diversity depending on how welcoming the organizers are and how connected the particular meetup is to a diverse pipeline.

5.1.5 Using social media

If you don't live in or near a city, there might not be any data science meetups near you. In this case, Twitter and LinkedIn are great places to start building your network. Once you follow a few well-known data scientists, you will frequently find more people to follow as you see who gets retweeted or mentioned a lot You can also start making a name for yourself. There are a couple of ways we like to use Twitter:

- **Share your work**: When you've written a great blog post or put an interesting analysis on Twitter, you want people to see it! It's totally normal to "self-promote" by linking to your work with a short description.
- **Share other's people work**: Read something great? A package saved you hours of frustration? Saw a slide in a talk that was particularly helpful? Help other people reach enlightenment too. In the next chapter, we'll discuss how one of the best ways to reach out to someone is to mention how you've benefitted from their work. If you tag the creator in your post, this is a great way to get on their radar in a positive way. If you are sharing a talk, check if there's a hashtag for the conference or meetup; using it in your tweet is a great way for it to have more visibility.
- **Ask for help:** Is there a problem you've been stuck on that you (and Google) haven't been able to solve? It's likely there's someone else out there who's faced the same issue. Depending on the type of problem, there may be specific forums or websites where you can ask questions, or you can put out a general call using a relevant hashtag.
- **Share tips:** Not everything warrants a blog post, but if you have a quick tip, share it. It may feel like something "everyone" knows, but remember, people just getting started don't. Even people using the language for years may not know of a new way to do something.

If you're able to be public about your job search, you can also post on social media that you're looking and ask if anyone has any leads. Even though you might not have a strong data science network yet, hopefully you have friends and former classmates and colleagues who might know of positions within their companies. This generally works better on social media where people connected to data science tend to congregate, such as LinkedIn or Twitter, but even social media networks like Facebook may have connections with opportunities.

It's common early in your career to feel like you have no network yet—networks seem to be held by people who already have data science jobs! The solution is to not only network when you're looking for a job, but long before it. The more you can get out of your comfort zone and talk to the people around you at places like conferences, meetups, academic institutions, BBQs, etc. the more prepared you'll be when it's time to look for a job the next time.

Keeping the pipeline full

A common mistake when job searching is to get your hopes pinned on one opportunity and not keep applying and interviewing elsewhere. But what happens if that opportunity falls through? You don't want to have to start the job search process over again from zero. You want to have multiple opportunities in each stage: applications you've sent out, Human Resources screens, take-home case studies, and in-person interviews. Don't consider the process over until you've accepted an offer.

Having multiple opportunities will also help you deal with rejection. Rejection is almost inevitable when job searching, and it's hard not to take it personally or as an indication of your worth. In some cases you may not even get notified you're rejected, you'll just never hear back. But there are so many reasons you might not have gotten the job that are

out of your control: maybe they closed the position without hiring anyone, went with an internal candidate, or accepted someone when you were still early in the process. Rejection will hurt, especially from somewhere you were really excited about, and you should take a little time to process your feelings. But having other options will help keep you motivated and making progress.

Finally, having multiple potential options also makes it easier for you to reject a job. Maybe you got through the HR screen and case study only to find there are no data engineers around, or the data science team is only a few people even though it's a large company, or what they're looking for is very different than they had advertised. While you shouldn't wait for the perfect data science role (as no such thing exists), you probably have a few non-negotiable requirements. But it's much easier to stick to those if you see they can be fulfilled in other jobs.

5.2 Deciding which jobs to apply for

You now should have a list of at least a dozen jobs you're somewhat interested in and could be a good fit for. Do you immediately apply for all of them?

Well, some people do apply to dozens or even hundreds of jobs. They're trying to play the odds, figuring that if there's a 10% chance of getting a response, applying to as many companies as possible will give you the most responses. But there's a fallacy there - if you have a finite amount of energy and time, spreading it across 100 applications instead of 10 will make each one weaker. We'll talk in the next section about how to tailor your applications to each position, but that's only possible if you're more selective in where you apply. And as we discussed, knowing people and having them refer you or at least getting an idea of the company from them is also the best way to get a first interview. It's pretty much impossible to do that for 50 companies. If you did try to meet that many people, you'd likely come off as insincere because you'd be looking to do it as quickly as possible.

> **R AND PYTHON** Should you apply for a job if they ask for Python and you know R, or vice versa? While knowing one language certainly makes it easier to pick up the other one, you'll already be learning a lot at your first data scientist job - working with stakeholders, the internal politics, statistics, the datasets, etc. Even if you could get the job, a new language on top of everything else can be difficult. Thus, we generally recommend only applying to jobs that use your main language. If knowing one of those languages is a "plus", but the other one isn't required, you probably want to be wary. That could mean that most of your work will not be coding. Finally, some jobs will ask for both. You also want to be a little wary here – usually that means that people use either one, not that everyone knows both, which can make collaborating difficult. This can work out but be sure to ask during your interviews how the split is – if you would be one of only two people using Python on the team of 20, it's going to be harder to improve your skills.

You'll want to return here to what you've learned in the first two chapters about the types of data science companies and of data science work. Do you want try all the different parts of data science, tuning a recommendation system one month and creating a lifetime value model the next? If so, you'll probably want to work at companies that have started doing data science more recently, as more mature companies will have specialized their roles. On the

other hand, big tech companies will also have legions of data engineers, so getting routine data is fast and easy.

Some of this will be obvious from the basic facts about the company – for example, a 10-person start-up is not going to have a mature data science system. But how can you find out more?

5.2.1 Researching the company online

First, see if the company has a data science blog. This will mostly only be tech companies, but these posts are invaluable to learn about what work they actually do and to include in your cover letter (covered in the next chapter). If you've never heard of a company, spend some time on their website. By knowing what the company does and how they make money, you can start making guesses at what kind of data science work they need. Finally, if you're really interested in a company, see if any of the company's data scientists have a blog where they talk about their work or have given talks about it.

When reading about a company, remember to also think about what's generally important to you. Does the ability to work from home sometimes matter? What about the number of vacation days? If you want to go to conferences, do they offer budget and time off? Reading what the company says about themselves can tell you their values. Do they talk about foosball, beer in the office, and catered dinner? That's likely to be a company full of young employees. Or do they emphasize environmental sustainability? Family leave? We'll discuss more in Chapter 8 how you can negotiate much more than salary, but for this stage, you can at least see if the company advertises benefits that align with your priorities.

Now that you have a manageable list of potential jobs, it's time to apply! In the next chapter, we'll walk you through creating a great resume and cover letter, including how to tailor them for each position.

5.3 Interview with Jesse Mostipak, Managing Director of Data Science at Teaching Trust

Jesse is the Managing Director of Data Science at Teaching Trust, an education non-profit. She comes from a background in molecular biology and worked as a public school teacher before falling in love with non-profit data science. You can find her writing on non-profit data science, advice for learning R, and other topics on her website, jessemaegan.com.

5.3.1 What recommendations do you have for starting a job search?

Think about how attached you are to the data scientist title. If you decide to not concern yourself with what you're called and to instead focus on the work that you're doing, you'll have a lot more flexibility to find jobs. Some non-data-scientist keywords to search for are "analysis," "analyst," and "data." While you'll have more to filter through, you may find a title like "research and evaluation" where you're qualified for that position but never would have come across it if you were just looking for data scientist.

When looking at jobs, focus on what you want to do as a data scientist. For me, I don't get a lot of pleasure from calculating the return on investment from website clicks. I asked myself, "what causes do I care about? Which organizations are aligned with that?" I cared a lot about the Girl Scouts, and they happened to be looking for an analyst position, so I was able to move in and do that. The same thing happened with Teaching Trust when I wanted to move more into education.

5.3.2 How can you build your network?

When I was making the transition to data science, I did a lot of things that failed for a very long time. I was the person who was reposting on Twitter every data science article that I saw, putting out 20 posts a day that had no engagement. You should think about who you want to meet, why, what value you bring to that relationship, and what's authentic to you. Think about branding yourself; not necessarily in a stringent way, but making sure how you show up online and in social media spaces is authentic to you. For me, I realized I couldn't be a perfect data scientist and wait to be on social media until I know all data science, because that was going to be never. I decided instead that I would talk about the things that I'm learning and be transparent about the process, and that's how I built my network.

5.3.3 What do you do if you don't feel confident applying to data science jobs?

If you're developing skills, can do some analyses in Python or R, and have the basics under control, you should focus on how you can get comfortable taking risks and failing. You have to fail a lot as a data scientist – if you are worried about taking a risk and failing in the job application process, what's going to happen when you take a risk on a model and the model doesn't work out? You need to embrace the idea of ambiguity and iteration. You have to apply and try; you're going to get rejected from jobs, but that's normal! I get rejected from jobs on a regular basis, it's just part of the experience.

5.3.4 What would you say to someone who is reading job postings and thinks, "I don't meet the full list of any job's required qualifications?"

Some research suggests that certain groups of people especially feel they need to be 100% qualified, whereas other groups say "I meet 25%? I'm applying!" Wrangle that confidence of the 25% qualified and go for it. But you may also be getting tripped up in decoding the language of the job description. For example, let's say there's a discrete skill listed, such as 10 years working with SQL databases. You might think, "I don't have that; I have 7 years working with Microsoft Access." But I would say that's still a transferable skill. It's on you as the applicant to tell yourself, "I may not have this exact skill, but I have one that's a lot like it. I need to take a look at SQL, see how transferable my skill is, and tell this company the amazing things I've accomplished with Microsoft Access and that they should hire me because I know I can do this with SQL and then some."

5.3.5 What's your final piece of advice to aspiring data scientists?

You need to develop your communication skills and your ability to "roll with it." You should be able to communicate across all levels of your organization in a way that respects the expertise of the people you're talking to and also shows you exist to make their lives easier.

By rolling with it, I mean being able to say something like, "That's not way I would approach that problem or this project, but I can see where you're coming from. Let's try it that way and maybe can I modify it in this way." You also need to be flexible because people are still just figuring out data science. Organizations want data scientists, but then they don't know what to do with them. You serve at the pleasure of your organization; if their needs have changed, you need to evolve and adapt to best meet those needs.

Finally, know that your job description might change. You have to be able to say, "this isn't what I thought I would be doing, but how can I make this work for me." You can't say, "I am the best at neural networks, but I'm not doing neural networks, so obviously this job is crap." You need to know that any position that you take is going to change and evolve as the needs of the company does.

5.4 Summary

- Search for general terms like "data" on job boards and focus on the descriptions, not the titles.
- Don't worry about meeting 100% of the qualifications listed.
- Remember the job search process is a two-way street: look out for red flags and think about what kind of data science you want to do.

6

The Application: Resumes and Cover Letters

This chapter covers

- Writing a compelling resume and cover letter
- Tailoring your application for each position

You've got a list of open jobs you're interested in; now it's time to let them know you exist! Pretty much every job will require you to submit a resume: a glorified list of your skills and past experiences. Most also ask for a cover letter too: a one-page letter describing why you should be considered for the job. It would be easy to quickly jot down your previous jobs and write a boilerplate letter saying you're interested in the company, but this is a situation where putting in more effort can be the deciding factor for making it to an interview.

In deciding what jobs to apply to, you've researched the companies to see if they would be a good fit. Now it's time to put that research to work by personalizing each application. In this chapter we'll start with making sure your base resume and cover letter are as effective as possible, covering best practices and common mistakes to avoid. Then we'll show how to take that "master" resume and cover letter and refine them for each job. Finally, you'll see networking can help accelerate your carefully crafted application into the hands of a hiring manager instead of an overflowing pile of resumes.

> **NOTE** the only goal of a resume is to convince a person who is barely skimming your resume and cover letter that you are a good fit

The key theme throughout this chapter is that you need to quickly convince a person that you are qualified for the position. Company recruiters often get hundreds of resumes for each

data science opening. Furthermore, because data science has so many distinct types of jobs within it, the range of skills of people applying to positions will be huge. This reinforces the notion that your materials have to say, "Hey, you reading this, you can stop skimming this massive pile because you found the person with the skills you were looking for." Being able to show you're qualified like that isn't an easy task.

While all of this is more work than spending an hour writing a basic cover letter and resume and applying with one click to dozens of jobs, it will yield much better results. You'll be more likely to get an interview, as you'll have matched your application to their requirements. And when you reach the interview, the topic of our next chapter, you'll be able to give a great answer to the common question, "Why are you interested in this role?"

6.1 Resume: the basics

The goal of your resume isn't to get you the job; it's just to get you an interview. Recruiters who run the interview process get in trouble if they bring in people who clearly don't meet the qualifications for the job, and they are praised when the people fit the qualifications well. Your resume needs to show the reader you meet the requirements for the position so that the recruiter is comfortable moving you on in the process.

That goal is very different from "create a catalog of every experience you had," which unfortunately is a goal that many inexperienced resume writers assume. While you do want to avoid having gaps on your resume by leaving off recent jobs entirely, you can spend less time on those not related to data science. And even if you have a lot of data science experience, you should still focus on highlighting the most relevant. If you have a multi-page resume, most recruiters won't have time to read all of it, nor will they be able to tell what parts to read. No one will tell you, "Well, we would have hired you, but you didn't put your job lifeguarding in high school on your resume, so we couldn't."

There will be plenty of time later in the interview process to go through your jobs, education, and data science projects in depth. For now, you want to focus on what's most relevant for meeting the qualifications of the position you're applying for. The rest of the process you'll be focused on your great qualities that will help you stand out beyond the other applicants, but for the first step, it's good to focus on fitting in to the hiring manager or recruiter's expectations.

With that in mind, let's walk through the basic structure of a resume, how to create good content within that structure, and how to think about the many "resume rules" that get thrown around. While plenty of this content could apply to any technical position in industry, as much as possible we will focus on what is unique about data science. We also made an example resume to use as a guide for what to expect.

SARA JONES

San Francisco, CA · 534-241-6264
sarajones@gmail.com · linkedin.com/in/sarajones · sarajones.github.io · github.com/sarajones

EXPERIENCE

JUNE 2019 – PRESENT, SAN FRANCISCO, CA
DATA SCIENCE FELLOW, AWESOME BOOTCAMP

- Built a web application in Python that recommends the best New York City neighborhood to live in based on someone's budget, lifestyle preferences, and work
- Analyzed 2,200 New York Times business articles (obtained via API) using natural language processing (TFIDF and NMF), visualizing how topics changed over time

AUGUST 2017 – JUNE 2019, SAN FRANCISCO, CA
INVESTMENT CONSULTANT, BIGCO

- Created a forecasting model in Python that boosted quarterly revenue by 10%
- Automated generating weekly market and industry trend reports

SEPTEMBER 2016 – JUNE 2017, NEW ORLEANS, LA
INTRODUCTION TO STATISTICS TEACHING ASSISTANT, COOL UNIVERSITY

- Led weekly review sessions of sixty students, earning a 4.86/5 rating in evaluations
- Created and open-sourced study guides that have been downloaded over 1,500 times

JUNE 2016 – AUGUST 2016, NEW ORLEANS, LA
ECONOMICS RESEARCH ASSISTANT, COOL UNIVERSITY

- Conducted an in-person experiment on decision-making with 200 participants, using cluster analysis to analyze the results in Python
- Published the resulting paper in the Journal of Awesome Economics

EDUCATION

JUNE 2017, NEW ORLEANS, LA
BA ECONOMICS, STATISTICS MINOR COOL UNIVERSITY
GPA 3.65/4.0
Relevant Coursework: Linear Algebra, Introduction to Regression and Statistical Computing, Experimental Design, Econometrics, Elements of Algorithms and Computation

SKILLS

- Python
- SQL
- Machine Learning
- Git
- Pandas
- Seaborn
- Scikit-learn
- NumPy

Example resume for an aspiring data scientist

6.1.1 Structure

In this section, we'll walk through each section of the example resume, going into more detail about what you want to include.

SECTION: CONTACT INFORMATION

SARA JONES

San Francisco, CA · 534-241-6264
sarajones@gmail.com · linkedin.com/in/sarajones · sarajones.github.io · github.com/sarajones

Having your contact information is necessary so that the recruiter can contact you! You need to put your first and last name, phone number, and email at a minimum. Beyond that, you can also put links to where they can find more information about you. These include social media profiles like LinkedIn, online code bases like GitHub, or personal websites and blogs. To figure out what to add, ask, "if someone clicks on it, would they think more highly of me?" For example, a link to your project portfolio from Chapter 4 is a fantastic thing to include. But a link to a GitHub profile that is empty save for a clone of a tutorial project is not. If you have any data science work publicly available, try and figure out a way to show it here.

Generally you also want to include the city and state you live in—this will let the recruiter know that either you are nearby and can commute to the job or that you would need to relocate if you got the job. Some companies are hesitant to relocate new hires because of the expense, so if you don't live nearby and don't want to bite that bullet you could potentially leave your location off.

If your legal name doesn't match the name you commonly go by, you can put your common name. Further along in the process you will need to let them know what your legal name is for things like background checks, but you aren't required to use your legal name when applying.

Finally, don't, use an email address that's potentially offensive ("i_hate_python@gmail.com") or something that might expire (like a school email).

SECTION: EXPERIENCE

EXPERIENCE

JUNE 2019 – PRESENT, SAN FRANCISCO, CA
DATA SCIENCE FELLOW, AWESOME BOOTCAMP

This is where you show that you're qualified for the job through previous jobs, internships, or bootcamps you've done. If they're related to data science, like software engineering, that's great: spend a fair amount of your resume on them. If a job isn't related to data science, like being an art history professor, you should still list the jobs but don't spend much time on it. For each position you've held, put the company name, the month and year of your start and

end, your job title, and at least one bullet (two or three for the most relevant jobs) of what you did. If you're a new or recent graduate, you can include internships and research jobs in college.

This section should be the largest in your resume and could potentially take up half the space available. It's also often the most important, since it's the first place recruiters will look to see if you have data science experience that could be related to the job they're hiring for. Due to the importance of getting this right, we'll go in-depth on how to create the best content for this part of the resume later on in the chapter.

SECTION: EDUCATION

In this section you list your different educational experiences, to hopefully show that you have a set of skills useful for the data science job. If you went to school past high school, even if you didn't get a degree, put your school(s), dates (same format as your work experience), and your area of study. If this will be your first job out of school and your GPA is high (above a 3.3) you can list it, otherwise leave it off. If you are a recent graduate and you took statistics, mathematics, or computer science classes, or any that involved these (like social science research methods or engineering classes), you can list those classes. Also, if you took a set of online data science courses you should include them here.

Recruiters will be very interested to see if you have an area of study that's relevant to data science, like a degree in data science, computer science, or math. They'll also be interested to see the level of the degree. Since many data science topics aren't covered until graduate levels of programs, having that will help. Recruiters generally won't care about what school you went to, unless it's extremely famous or prestigious, and even then that won't matter if you're more than a few years out of school. It's nice for recruiters to see any bootcamps, certificates, or online programs since it shows you've furthered your education.

While the education section of your resume can give valuable information to the recruiter, you can't really improve the section by doing anything less than going out and getting an additional degree or certificate, which we've covered in Chapter 3.

SECTION: SKILLS

This section is where you can explicitly list all the relevant skills you have to contribute in a data science setting. Ideally a recruiter will see this section and nod while saying, "yes, good" since you'll have skills listed that are relevant for the job. For data science resumes there are two types of skills to list here. The first type is programming/database skills which can be programming languages like Python and SQL, frameworks and environments like .NET or JVM, tools like Tableau and Excel, or ecosystems like Azure and AWS. The second type is data science methods like regressions or neural networks. Since there are so many possible methods you can list, try to focus on a key few that together show you have the essentials and a few special skills. For example, something like: "regressions, clustering methods, neural networks, survey analysis" shows you have the basics and depth. Try not to list more than 5-6 to avoid overwhelming people, and don't list skills that have no chance of being relevant for the job (like an obscure academic programming language from your time in grad school).

Only put skills on that you would be comfortable using on the job, not a language you haven't touched in five years and don't want to pick up again. If it's on your resume, it's fair game to ask about it. If there are skills you've seen requested in the data science job postings you've viewed and you have them, make sure to list them! That's exactly what the recruiters will look for.

We recommend that you don't use star ratings, number, or another method to try to indicate how strong you are in each skill. For one, it doesn't actually mean anything: anyone could put a 5/5. If you give yourself all perfect scores, a hiring manager may think you're not honest or good at self-reflection; if you give yourself a lower score though, they may doubt your abilities. It's also not clear what you consider each level to be: does a 5/5 mean you think you're one of the best in the world, that you know how to do an advanced task, or that you're better at it than your co-workers? If a hiring manager does want you to self-assess your level of different skills, they'll ask you in an interview.

Don't put soft skills like "critical thinking" or "interpersonal skills;" while these are crucial for being a successful data scientist, saying those on your resume is meaningless since anyone can just say them. If you do want to highlight your skills in these areas, talk about how you utilized them in specific instances within the experience section of your resume. You also don't need to list basic skills anyone applying would be expected to have, like "Microsoft Office Suite."

SECTION: DATA SCIENCE PROJECTS (OPTIONAL)

If you have data science projects you've done outside of work, you can create a section for them. This is great for candidates who have less work experience but have done projects on the side or in a school or boot camp. You're basically telling the recruiter, "While I may not have much relevant work experience, that doesn't matter because I've still done the full data science process."

For each project you'll need a title, a description of what you did, how you did it, and the results. In fact, the data science projects should look as though they are jobs in structure and content, so everything in the section below on generating content applies for these too. Ideally you have a link to a blog post or at least to a GitHub repository that has an informative README. Data science is a technical field where it's unusually easy to just show the work you did, and here is a great place to do it. If you have enough relevant work experience, you can skip this section and still talk about projects in your interviews.

SECTION: PUBLICATIONS (OPTIONAL)

If you published papers in a master's or PhD program that are related to data science, you should include those. If you published papers in other fields, even quantitative ones like physics or computational biology, you can include them but only briefly. Since they are not directly related to data science the person reading them won't get much out of the publications except "you worked hard enough to get a publication." You can list the relevant work you did during your research in the experience section, like "created an algorithm to analyze millions of RNA sequences per minute." But a publication in a journal the hiring manager has never heard of, even if it's a prestigious one in your field, won't go too far on its own.

OTHER SECTIONS

You can add other sections, like "honors and awards" if you've won Kaggle competitions or received a scholarship or fellowship, but they aren't necessary. You don't need to include references; speaking to your references will come later in the process and you can share that information if you progress to there. Objective statements are usually not needed and redundant given the other information in your resume. For example, "Data Scientist experienced in Python looking for a position to develop A/B testing and modeling skills" is not going to make a recruiter more excited!

PUTTING IT TOGETHER

Generally, you put your contact information at the top, followed by the next most important section. If you're in school or just graduated, that's probably your education; if you don't have relevant work or education, put your data science projects at the top; otherwise, put your

work experience there. Within your work and education sections, put your experiences in chronological order, from most recent to least.

We've seen lots of different effective formats for data science resumes. In this field you have a bit from freedom in your design; there is nothing close to a standard format. But with that freedom you always want to focus on making your resume easy to scan quickly. Since recruiters spend so little time looking at your resume, you don't want them to spend that time trying to figure out how to find your most recent job! Don't make your aesthetic design distract from your content; constantly consider how others will view it. Some good practices include:

- Clear headers for each section so it's easy to jump between them.
- More whitespace to make it easier to read the content.
- Bold important words like the titles you've held at each company.

If ideas like whitespace and headers are overwhelming, stick to a resume template from online or consult a design specialist.

You want to limit your resume generally to a single page. This serves two purposes. One, given the brief skim it'll get, you want to make sure they're spending that time on the information you think is most valuable. Second, it shows you can communicate concisely and understand what parts of your experience are most important to share. If a person submits a 17-page resume (which we've seen), then it strongly suggests they have no idea what in their past makes them a good candidate, and that they feel entitled to other people's time for them to read it.

Make sure you're consistent throughout your resume. If you abbreviate month names in your education section, abbreviate them in your work experience section too. While you can have different fonts and sizes for headings and body, don't switch it up from bullet to bullet. Use the past tense for previous positions, present for a current one. These things show that you pay attention to the smaller detail and again help the reader to quickly process your content, as they won't be distracted by font or style changes. While a single inconsistency is unlikely to cost you an interview sometimes details make all the difference.

> **PROOFREADING** It's essential to proofread your resume! A few typo or grammatical mistakes may lead your application to the (metaphorical) trash-bin. Why so harsh? When sifting through hundreds of resumes, two kinds stand out: those that are clearly exceptional (rare) and those that are easy to eliminate. The latter needs some rules of thumbs, and in addition to ones about clearly not meeting the requirements, typos are an easy one. Data science jobs require attention to detail and checking your work; if you can't do that when putting your best foot forward in an application, what does that suggest about your work? In addition to using spell check, have at least one other person carefully read over your application.

6.1.2 Deeper into the experience section: generating content

Hopefully, coming up with the dates and titles of your work and education history are easy enough to fill out. But how do you come up with those punchy bullet points to describe your work experience (or data science projects)?

The common mistake people make on their resume is to just create a list of their job duties, like "generated reports for executives using SQL and Tableau" or "taught Calculus to three sections of 30 students." There are two problems with this. One, it's just stating what you were responsible for, not what you accomplished or how you did it. Second, it may not be framed in a way that's relevant to data science. So for the previous two examples, you could describe the same work as "automated the generation of sales forecasts reports for executives using Tableau and SQL, saving 4 hours of work each week " or "taught Calculus to 90 students, earning an average of 9.5/10 in student evaluations and 85% of students getting a 4 or 5 on the BC Calculus AP Exam."

As much as you can, you want to explain your experience in terms of transferable skills to data science. *Even if you haven't worked in data science or analytics, was there any data that you did work with?* Hiring managers are willing to consider experiences outside of data science roles as still relevant, but you have to explain why they should. If any of your work can conceivably be related to taking data and understanding it, you should put enormous effort into making a concise story on what you did. Did you analyze 100GB of star data for an astrophysics PhD? Did you manage 30 Excel files to plan staffing for a bakery? There are lots of activities that are using data to understand a problem.

Have you used tools like Google Analytics, Excel, or Survey Monkey? Even if those may not be the tools the current job is asking for, working with data of any type is relevant. What communication skills did you use? Have you explained technical or niche concepts, maybe in PhD research talks or to other parts of the business? If coming up with transferable skills is difficult, don't worry – the rest of the advice on writing better bullets will still help. But if you haven't already, you should think about how your education or side projects can demonstrate data science skills, especially if your work experience can't.

For the least relevant positions that were a few years ago, it's okay to just have one bullet. You generally don't want to leave off a job though if it will leave a gap in your resume of more than a few months. If you've been in the workforce for a while and had a lot of jobs, it's okay just to list the three or four most recent.

You might be finding this process is a lot easier for the job you're currently in than the one you had five years ago. One good practice is to keep a list of accomplishments and the major projects you've worked on. When you're in it each day, making incremental progress, you can forget how impressive the whole is when you step back. People know that your resume isn't an exhaustive list, so they won't think, "It took her 15 months to build an automated system for tracking and scoring sales leads that saved their sales team over 20 hours of manual work a week?" They'll think, "Wow, we need a system like that!"

Generally, bullets can fall into two types. The first is big accomplishments; for example, "Created a dashboard to monitor all running experiments and conduct power calculations." The second is average or totals, such as "Implemented and analyzed over 60 experiments, resulting in more than $30 million in revenue."

In either case, your bullets should each start with a verb and are ideally quantifiable. Rather than saying, "I made presentations for clients," write, "created more than 20 presentations for Fortune 500 executives." It's even better if you can quantify the impact you had; writing "ran 20 A/B tests on email campaigns, resulting in a 35% increase in click rate and 5% increase in attributed sales overall" is much more powerful than "ran 20 A/B tests on email campaigns."

6.2 Cover letters: the basics

While the purpose of a resume is to give hiring managers relevant facts about your work experience and past, the purpose of the cover letter is to help them understand who you are as a person. Your cover letter is where you can explain how you've researched the company and highlight why you are a great fit. If your resume doesn't show a linear path, a cover letter can pull that together and explain how all the pieces fit to make you a great candidate for this job. Even just showing that you know what the company is, that you've read their about page, or that you've used their product (if it's available for consumers for free) goes a long way. Your letter is your best tool to help hiring managers understand things that don't fit well in bullet lists.

Unlike a resume, a cover letter may be optional. But if a company has a place to submit a one, do so—some companies will eliminate candidates if they haven't written one. It's not uncommon for companies to even give a specific thing for you to write about, like your favorite supervised learning technique; this usually is to check whether people actually read and followed the request of the job description instead of just sending out a generic cover letter everywhere. You definitely want to let the company know that you can follow instructions.

Knowing that a cover letter is to help the company understand who you are better, a common mistake we see in cover letters is focusing on what the company can do for you. Don't say "this would be a great step for my career." A hiring manager's job is not to help as many careers as possible, it's to hire people that help the company. Show them how you can do that. Even if this would be your first data science job, what relevant experience do you have? What record of achieving results, even if they're not data-science related, can you share so it's clear you work hard and accomplish goals? Don't undercut yourself; try and think broadly as to how you can make yourself appealing to the company.

Just like your resume, you want to keep it short; three-quarters to one page is usually the rule. Focus on your strengths. If the job description lists four skills and you excel in two, talk about those two! Don't feel like you have to make excuses for skills that you lack. Below is an example cover letter we're using as a discussion point.

SARA JONES

New York, NY · 534-241-6264
sarajones@gmail.com · linkedin.com/in/sarajones · sarajones.github.io · github.com/sarajones

GREETING

Dear Jared,

INTRODUCTORY PARAGRAPH

I am writing to express my strong interest in applying for the Data Scientist position at Awesome Company. I've enjoyed reading Awesome Company's data science blog since it started 8 months ago. The post on using topic modeling to automatically generate tags for your support articles was immensely helpful in one of my own projects to classify articles in the New York Times business section.

1-2 PARAGRAPHS OF DATA SCIENCE WORK EXAMPLES

I recently graduated from Awesome Bootcamp, a full-time, 3-month Data Science immersive. At Awesome Bootcamp, I designed, implemented, and delivered data science projects in Python involving data acquisition, data wrangling, machine learning, and data visualization. For my final project, I gathered 3,000 neighborhood reviews and ratings from Neighborhood Company. By using natural language processing on the reviews and available listings from Real Estate Company's API, I built a recommendation system that will match you to a neighborhood based on your budget, preferences, and a free-text description of your ideal neighborhood. You can try it out here: myawesomewebapp.com.

Prior to Awesome Bootcamp, I was an Investment Consultant at BigCo. When I joined, my team of six was all using Excel. While exceeding my targets, I began automating common tasks in Python, such as generating a weekly market and industry trends report, saving the team hours each week. I then developed a tailored curriculum to teach them Python. The initiative was so successful the company asked me to develop a full 2-day workshop and flew me out to three other offices to teach it, reaching over 70 consultants.

CLOSING PARAGRAPH

I am confident that my expertise in Python, academic training in Economics and Statistics, and experience delivering business results would make me a great fit for the Data Science team. Thank you for your consideration.

SIGN OFF

Sincerely,
Sara Jones

An example cover letter with highlights showing the different components

6.2.1 Structure

Cover letters have a less well-defined set of rules than resumes. That being said, here's a good general structure you can follow:

- **Greeting**: Ideally, find out who the hiring manager or recruiter is for the position. The first place to look is the job description itself; the hiring manager's name might be listed. Even if it's just their email address, you can probably figure out who it is with some googling. Otherwise, check LinkedIn and the company website to see if one of them identifies the team leader. Even if you end up too high, maybe at the VP of the department who will be your skip-level (manager's manager), it still shows you did your research. Finally, if you can't find a name, address your letter to "Dear Department Name hiring manager," e.g. "Dear Data Analytics hiring manager." Don't say "Dear Sir or Madam" – it's archaic and generic.
- **An introductory paragraph**: introduce yourself, name the position you're interested in, and briefly explain why you're excited about the company and role. If the company has a data science blog or if any of their data scientists have given talks or written blog posts on their work, this is a great place to mention you've read that. Connect what you learned there to why you're excited about the position or company.
- **One to two paragraphs with data science work examples**: connect your previous accomplishments to this role. Add details to something you discussed in your resume. Pick one role or side project and go in depth. You want to give specific examples. Follow the "show don't tell principle;" instead of writing that you're a "detail-oriented, organized problem-solver," give them an example of when you did this at work.
- **A closing paragraph**: thank them for their time and consideration. Sum up your qualifications in explaining why you'd be a good fit.
- **A sign-off**: "sincerely," "best," and "thank you for your consideration" are all good closings; avoid anything too casual or overly friendly, like "cheers" or "warmest regards."

6.3 Tailoring

The previous two sections laid out general rules for an effective cover letter and resume. But the biggest way you can differentiate yourself is by closely tailoring them to the position you're applying for.

The first person screening your data science resume is likely not the manager for a position; it may not even be a human! At larger companies, they'll usually have applicant tracking systems that automatically screen resumes for keywords, surfacing those that have it. This system might not recognize "linear modeling" as meeting the requirement for experience in "regression." A human reader might not either; an HR person may have been given nothing besides the job description and instructions to find promising candidates. You don't want to risk a recruiter not understanding that your project using "*k*-nearest neighbors"

means you have experience in clustering analysis, or that "NLP" is the acronym for "natural language processing." You want someone to be easily able to look back and forth between your resume and the job description, finding exact matches for the requirements in your experience. While you don't want to overload your resume with tech jargon, you do want to have the key words (like R or Python) in there a few times to make it through these screens.

We recommend you have a "master" resume and cover letter you can pull from rather than starting from scratch. This is especially helpful if you're applying for different types of positions. If some emphasize machine learning and others exploratory analyses, it's much easier if you have bullets and key terms related to each ready to go. Your "master" resume and cover letter can be longer than one page, but make sure your ending result is always under a page.

Tailoring your application to the position doesn't mean you need to have one bullet or skill for every single requirement. As we discussed in the previous chapter, job descriptions are generally wish-lists. Try to figure out which ones are the core skills for the job; sometimes companies will helpfully divide skills and experience into "requirements" and "nice-to-haves," but even if they don't, you might be able to tell by the description of the job responsibilities. While companies would love to get someone who gets a check-plus on everything, most won't be holding out for it.

One exception is big tech firms and well-known, fast-growing start-ups. These companies get a lot of candidates and are looking for reasons to reject people. They're very worried about false positives, meaning hiring someone who is bad or even just average. They don't really care about false negatives, not hiring someone who is great, because they have lots of great people in the pipeline. For these companies, you usually do have to meet 90% if not 100% of the requirements.

6.4 Referrals

Company websites and job boards all have a place you can apply, sometimes even just with a click of a button if you've saved your resume on the job board. Unfortunately, because it's so easy, your resume will often end up in a pile of hundreds or even thousands of other similar "cold" applications. That's why we recommend avoiding applying this way until you've exhausted other options. Reading job postings is a great way to get a feel for what kind of jobs are available, but the best way to get your foot in the door is to have someone hold it open for you.

You want to try to use the hidden back door to most companies: referrals. This is when a current employee recommends someone for a position, usually by submitting their application and information through a special system. Many companies offer referral bonuses, where employees can get a couple thousand dollars if they refer someone who gets and accepts a job offer. Companies like people who are referred because they come "pre-vetted": someone who already works at the company and (presumably) is doing well thinks this person would be a good fit. Even if it's not that formal, being able to write on your cover letter "I discussed this

position with [Star Employee X]" and to have that person tell the hiring manager to look out for your resume is a huge benefit.

How do you find people who can refer you? Start by looking at LinkedIn and seeing if you know anyone who works at a company you're interested in. Even if you haven't spoken in a while, it's perfectly fine to reach out with a polite message. Next, look for people who previously worked at the same company or went to the same school as you. You're more likely get a response from a cold message if you mention something you have in common. Finally, look for people who are "second degree" and see who you have in common. If you're on good terms with any of your mutual connections, reach out to that connection to see if they'd be willing to introduce you.

If you're reaching out to a data scientist, take some time to learn about what they do. Do they have a blog, Twitter, or GitHub where they've shared their work? Mark Meloon, a Senior Data Scientist at ServiceNow, wrote in his blog post "Climbing the relationship ladder to get a data science job" about how the most effective messages would be ones that combine a compliment to the content he's published with a request to ask some more questions. This way, you'll also avoid asking them about things they've already publicly talked about and can focus on getting advice you couldn't find elsewhere.

Remember that it's not only people in data science who can help you. While other data scientists will be best positioned to tell you what it's like to work at their company, people in any position can refer you. If someone you know works at a company you want to apply to, reach out to them! At the very least, they can still offer you insight into the company culture.

CRAFTING AN EFFECTIVE MESSAGE

In his blog post, "Do you have time for a quick chat?", Trey Causey, Senior Data Science Manager at Indeed.com, outlines some suggestions for effectively reaching out to someone you don't know to talk about your project, job search, or career choice. By following these guidelines, you'll be much more likely to get a response, have a productive meeting, and build a good foundation for a continuing relationship.

1. Have an agenda for what you want to discuss and include it in your email.
2. Suggest a few times (including an ending time 30 minutes later) and a location near the person's work.
3. Buy their lunch or coffee.
4. Get there early.
5. Have specific questions and goals for your conversation based on the agenda you sent. Don't just ask for "any advice you can give me."
6. Keep track of time and let them know when you've used up all the time you asked for; if they want to keep talking, they will.
7. Thank them and follow-up on anything you've talked about.

Here's how Trey pulls that all together into a sample message:

"Hi Trey, I read your blog post on data science interviews and was hoping I could buy you a coffee at Storyville in Pike Place this week to ask you a few questions about your post.

I'm currently interviewing and the part about whiteboard coding was really interesting to me. I'd love to hear your thoughts on how to improve whiteboard coding questions and answers as well as share some of my own experiences with these types of questions.

Could you spare 30 minutes sometime, say Tuesday or Wednesday of next week? Thanks for writing the post!"

6.5 Interview with Kristen Kehrer, a data science instructor and course creator

Kristen Kehrer is a Data Science instructor at UC Berkeley Ext, Faculty at Emeritus Institute of Management, and Founder of Data Moves Me, LLC. Data Moves Me helps data science teams communicate machine learning model results to stakeholders so that the business can confidently make decisions. Kristen holds an MS in Applied Statistics and is co-author of the book "Mothers of Data Science."

6.5.1 How many times would you estimate you've edited your resume?

Oh, a million! I come from a blue-collar family, where my dad was a firefighter and my mom stayed at home, so I was never taught how to write a great resume for industry. But I did okay by asking others for help when I was getting out of grad school. I also have always been the type to keep track of any new project I work on or anything interesting that I could add to my resume. I wasn't one of those people who'd go for two years without updating my resume. More recently, my old company paid for a career coach when they laid me off. I got to learn all about resume best practices and how to effectively position myself to land a great job.

I absolutely advise people to update their resume often. Especially if you've been working at the same place for a while, it is very difficult to try and think about all the relevant things that you could add to your resume. For example, I co-authored a couple posters in the healthcare industry that won awards. That's not relevant to every position I apply for, but if I am applying to a position in healthcare, I'd want to be able to reference that research. If I didn't keep track of it, I would not be able to remember who my co-authors were or what the title of the poster was.

6.5.2 What are common mistakes you see people make?

So many things! One is the four-page resume that still has that they were a swimming coach. Another is not realizing that the applicant tracking systems don't parse certain things well. If people have icons or charts on their resume, that's going to come through as a blob on a lot of the older automated systems and may end up with you being automatically rejected. I also don't like when people put, say, three stars for Python, because you're not giving people any context and whichever skill you're putting two stars for you're saying that you're not good at that thing.

6.5.3 Do you tailor your resume to the position you're applying to?

I'm not obsessive about it. But almost all medium to large companies now use an applicant tracking system and you want to be able to rank high in terms of matching key words. If I saw things on a particular job description that matched things that I've done, but were maybe worded slightly differently, I'd just edit a couple words to match the verbiage that they're using on their job description.

6.5.4 What strategies do you recommend for describing jobs on a resume, especially for people who haven't had data-related jobs before?

I tell people to optimize their resume for the job they want, not the job they have. You don't need to make a list of all the things that you've ever done. Instead, think about what you've done that you can reposition for data science. For example, if you're a math teacher, you've been explaining technical or mathematical material to a non-technical audience. Or maybe you worked on a project where, even though it wasn't in analytics, you had to work cross-functionally across multiple teams. Overall, you want to be able to show that you're able to solve problems, self-manage, communicate well, and achieve results. Finally, you can use side projects to highlight your technical chops and the initiative that you're taking.

6.5.5 What's your final piece of advice for aspiring data scientists?

You need to start applying to data science jobs. Too many people just keeping taking online courses because they think they need to know a million things to become a data scientist, but the fact is, you're going to start a job and you're still going to have more to learn. Even ten years in I still have more to learn. By applying, you'll get feedback from the market. If nobody responds to your resume, maybe it's that you're not positioning yourself well, or maybe it's that you don't quite have the skills. Gather some feedback from a few people and then choose an area to focus on, like being able to automate processes in Python. Work on that, add it to your resume and apply to more. You need to apply, get responses, and iterate and move forward until you get a job.

6.6 Summary

- Your resume is not an exhaustive list of everything you've ever done. You need it to get you an interview, not the job, so focus on matching it closely to the job description.
- Cover letters let you show why you're interested in an organization and how your experience positions to make a valued contribution.
- Talking to people who currently work at a company, especially if they're data scientists, is the best way to get insight into specifics about openings and hiring manager priorities.

7

The interview: what to expect and how to handle it

This chapter covers

- Delivering what interviewers are looking for
- Common types of interview questions
- Proper etiquette when communicating with a company

If you stop and consider the process of an interview you might realize just how tricky it is: somehow you need to show total strangers that you would be good at a role you only know about from a few paragraphs in a job posting. In the interview they may ask you technical questions of all different levels about all different technologies—some which you may not have even used before. Further, during the interview you'll need to learn enough about the company to be able to decide if you'd *want* to work there. You will have to do all of these things in only a few hours, all while acting professional and proper. It's enough to give you some serious anxiety sweats.

The good news is that with the right preparation and mindset, data science interviews can be taken from "panic attack inducing" to "manageable" or "tolerable" or perhaps even "an enjoyable experience." In this chapter we will walk through what interviewers are looking for during interviews and how to adjust your thinking to align with their needs. We'll walk through what you should expect from the technical and non-technical questions, as well as a data science case study. Lastly, we will go over how to behave and what questions you should be asking interviewers. With this, you should be well prepared for what lies ahead.

7.1 What do companies want?

When employees of a company are interviewing candidates for an open position, they are looking for one crucial person:

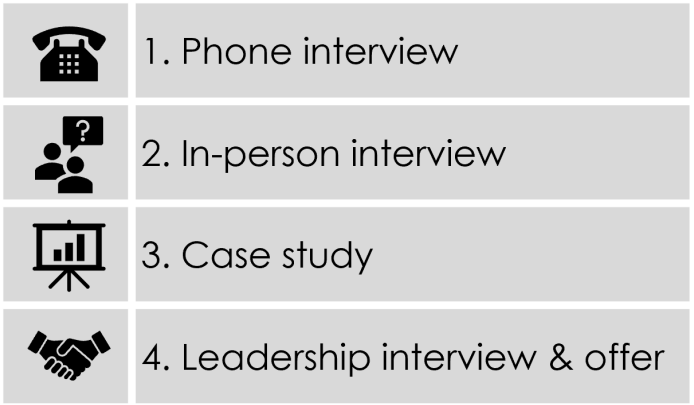
Someone who can do the job.

This is the only type of person they are looking for. Companies are not looking for the person who get the most interview questions "right" or who has the most degrees or years of experience. They only want someone that can do the work that needs to be done and help the team move forward with their goals. But what does it take to do a job? Well, a few things:

- **Having the necessary skills.** The necessary skills can be both technical and non-technical. On the technical side, this means understanding the skills covered in Chapter 1: some combination of math and statistics as well as databases and programming. On the non-technical side this is general business acumen, as well as things like project management, people management, visual design or any other number of skills depending on the role.
- **Being reasonable to work with.** If say something offensive, act defensive, or have any other number of character flaws that would make it difficult for other people to interact or collaborate with you, a company will not want to hire you. This means that during the interview (and always, actually) you'll want to be agreeable, compassionate and positive. This does not mean that your interviewer should want to "grab a beer with you," since that kind of filtering often creates homogeneous culture which is bad for companies. It just means that the people on your future team need to see you as someone they want to work with.
- **Being able to get things done.** It's not enough to just have the skills to do the job, you have to actually use them! This means finding solutions to problems on the job and implementing those solutions. Data science has lots of places where a person can get stuck: figuring out messy data, thinking through the problem, trying different models, and tidying a result. A person who can overcome each of those challenges will be much better at doing the job then someone who sits around waiting for help without asking for it. People who try to make everything perfect have trouble with this as well—never being able to call your work done means you can never put the work to use.

Knowing that those three things are what companies look for in candidates, let's jump right into the interview process. As we walk through how the interview works and the questions that come up often, we will frame our discussion in terms of those three ideas.

7.2 The interview process



The four steps of the interview process

While the exact process of a job interview varies by company, they all tend to follow a basic pattern. This pattern is designed to maximize the amount of information a company learns about the candidate, while minimizing the amount of time it takes for people within the company to run the interview. The people who do interviews are typically busy, have many interviews to facilitate, and want to allow for fair comparisons between candidates, so the process is streamlined and consistent. Here is a basic outline of what to expect in an interview process:

1. **An initial phone screening** – this will typically be a 30 (or sometimes 60) minute phone interview with a technical recruiter: a person who has lots of experience in screening candidates and knows technical terms, but doesn't do technical work themselves. From the company's perspective, the goal of this interview is to check if you have any chance of being qualified for the job. The recruiter wants to screen out people who clearly wouldn't be a good fit. This includes people who aren't remotely qualified (don't have the required skills) or who sound abrasive or mean (and won't work well with others). From a technical side, the interviewer is just checking that you potentially have the minimum skills needed, not that you're the best at them. They're much more likely to ask, "have you used linear regressions before" than, "how do you compute the maximum likelihood estimate for a gamma distribution." After the first phone interview, the company occasionally will have another phone interview with a more technical screener. If the initial phone screening goes well, within a few weeks you'll have:

2. **An on-site interview** – this is often 2 to 6 hours and makes up the main part of the interview process. During this visit you'll get to see where you will work and meet the people who you would work with. This interview provides the company time to ask more probing questions about your background, your skills, and your hopes and dreams as a data scientist. You'll be interviewed by multiple people during the visit, each asking questions on different topics, some technical and some non-technical. The goal of this interview is to ensure (via technical questions) that you have the necessary skills and (via behavioral questions and how you conduct yourself) that you are reasonable to work with. If this interview goes well then it's time for:
3. **A case study** – you'll be given a description of a real-world problem and data related to it. You'll either be given time on site or a weekend at home to analyze the data, try and solve the problem, and to create a report around it. You'll then have to present your report to the hiring team. This exercise shows the team that: you have the necessary skills (through how good your report is) and that you can get things done (through how much you were able to do in the report). Not all companies do this step, and sometimes they replace it with having you do a presentation on past work. If your case study report goes well you'll then have:
4. **A final leadership interview and offer** – this interview is with the senior manager, director, or some other leader of the team. The purpose is for the leader to give their approval of your fit for the position and the team. If the leadership interview even happens in the first place it means the data science team thought you were a good fit so it's rare for this interview to overrule their approval. Note that this interview often happens right after the case study, but also could happen at the beginning or end of the first on-site interview. Assuming it goes well you'll get an offer in less than two weeks!

With these steps it usually takes somewhere between three weeks and two months for you to get an offer letter after submitting your resume. As you can see, each part of the interview process is designed to achieve a different goal for the company. Let's dive into each part of this process and to showcase your skills and capabilities in each setting.

7.3 Step one: the initial phone screen interview

Your first interaction with the company will likely be a 30-minute phone call with a recruiter. Since it's the first interaction it's important to make a good impression. However, depending on the size of the company the person you'll be talking to is likely unrelated to the data science team you'll be working on, so y*our goal is to show the company that you are a person who could do this job, not necessarily that you're the best person for the job.*

Why? Because the person you will be talking to in this phone interview has the job of filtering out the unqualified candidates. When the recruiter talks to a candidate, they are trying to assess if it's worth it for someone on the data science team to talk to the candidate. Lots of times people apply to jobs they don't have the skills for (or they lie about having the required skills), and so the recruiter wants to prevent them from moving onward. As a

candidate, your goal here is to get the recruiter to understand that you are at least minimally qualified for the position.

The recruiter is likely to ask you these sorts of questions.

- "Tell me about yourself" – this is the recruiter asking for you to give a 1-2 minute overview of your background. They're looking to hear you describe previous experiences that relate to the job at hand. For example, if you're applying to a decision science position, they'll want to hear about your academic background and any jobs or project where you did an analysis. It's important that your answer falls in the 1-2 minute range. If your answer is less than a minute you'll seem like you don't have enough going on, and if it's over two minutes you'll seem like you don't know how to summarize a story.
- "What sorts of technologies are you familiar with" – this is the recruiter checking if you have the technical experience to do the job. Beyond this specific question, you should expect to be asked questions around math and statistics, databases and programming, and the business domain, just like from Chapter 1. You'll also want to list any technologies you know that are related to the position. In the case of data science that's tech like R/Python, SQL, and cloud technologies like AWS or Azure. If you don't have exactly what the job posting asked for (like Python instead of R), that's okay; just be open and honest about it. If you happen to know the tech stack the company uses, try and frame your answer around that.
- "What makes you interested in this position" – this is the recruiter trying to understand what drew you to the company in the first place. A particularly well thought out answer shows you do your homework and can get things done. Answering "I just clicked 'apply' on every data science job on LinkedIn" would show you have poor judgement. Don't overthink questions like these; just demonstrate that you know what the company does and have a genuine interest in the role. As much as possible try and connect the role to your background and interests.

While the recruiter will ask you questions about you and your background on this call, it's also a time for you to get a better understanding of the position itself. The recruiter should spend at least 10 minutes on the call talking more about the role and the team you are interviewing for. During this time, it's important to ask questions to the recruiter so you can be sure you'll want the job and so that you can demonstrate a sincere interest in the role. These questions can include topics like travel, the company's culture, how the team is changing, the team's priorities, and why the role opened up in the first place.

It is possible that on the call the recruiter will try and understand your salary expectations. That could be either directly (e.g. "what are you expecting for a salary?") or indirectly (e.g. "how much do you currently make?"). In a positive light, this is the recruiter making sure that the company can afford to meet your salary expectations, so as to not waste time interviewing someone who wouldn't take what the company can offer. In a negative light, this could also lead to the recruiter trying to lock you into a salary that's lower than they would potentially

offer since you're giving them information about your expectations. As much as possible avoid discussing salary until you're further in the process. *We'll cover salary negotiation in the next chapter on negotiating the offer.*

During the call, make sure to ask what the next step in the process is, and what the timeline is. The recruiter should tell you something like "we would bring you in for an interview next, and we should know in a week if that will happen or not." This is a totally normal and fine thing to ask. Don't ask the recruiter directly if you are going to move on to the next interview. That will put the recruiter on the spot and possibly make them feel uncomfortable—and they likely don't have the authority to make the decision anyway.

If the phone interview goes well, you'll be asked to move on to an on-site interview.

7.4 Step two: the on-site interview

This step is the heart of the interview process. The company invited you on site to do an interview and booked it for multiple hours. You took a day off work, you put on some nicer-than-normal clothing, and you are heading out.

What to wear to the interview

One topic with interviews that is often discussed and debated is what to wear to it. For data science positions, this is complicated by the fact that the jobs can fall in all sorts of industries, each with its own culture and dress code. What is completely appropriate for one interview may be totally inappropriate for the next.

Your best bet is to ask your recruiter when scheduling the interview both what people wear to interviews with the company, as well as what the general company dress code is. The recruiter wants you to succeed and shouldn't lead you astray. Otherwise try to talk to someone who works at the company or a similar company. If all else fails assume that more bureaucratic industries have strict dress codes (finance, defense, healthcare), whereas young companies or tech companies have loose dress codes (startups, massive tech companies). Avoid extremes (sandals, shorts, cocktail dresses, top hats), and wear something you feel comfortable in.

The goal of this interview is for the company to understand if you would be able to do the job they are hiring for—and if you would do it well. Depending on the company and position they may have anywhere from three to ten different people coming in for an interview, each with their own strengths and weaknesses. The company wants to find the person who would be the best fit, or rather they want to find the first person they interview who would be pretty good at it.

That means that throughout the interview you'll be wanting to reinforce the idea that you can do the job well. That's different from being the smartest candidate, the candidate with the most years of experience, or the candidate who has used the most types of technology. No, you want to be a candidate who has a healthy balance of being reasonable to work with, has enough skills to do the work, and can get things done.

What will happen during these multiple hours of interviews? An on-site interview usually includes the following:

- **A tour of the workplace and an introduction to the team.** The company wants you to get an understanding of what working with them will be like, and potentially impress you with their free beverages and snacks (if available). This will take less than 15 minutes, but it's good to get a decent look around to see if you would be happy working there. Are the workspaces calm and easy to work in? Do the people look reasonably happy and friendly? Are the laptops eight years old? When you are walking around on this tour you'll often be making small talk with someone from the company. Watch out! This is part of the interview—if you come off as unpleasant or mean, that could affect the job-offer decision. Shoot for coming off as nice but most importantly, be your authentic self.
- **One or more technical interviews.** This can be anywhere from 30 minutes to multiple hours depending on how rigorous the company is. You'll be asked questions covering many topics and may need to do work on a whiteboard or computer. We'll go into more detail about these questions later in the chapter. The point of this interview isn't for the interviewer to understand what deepest topics you know, or if you can solve the trickiest problems. The point is to see if you have the minimally necessary skills to do the job. So your objective is to show you have what is needed.
- **One or more behavioral interviews.** The point of behavioral interviews is to understand how well you get along with others, and how well you get things done. You'll be asked a lot of questions about your past experiences, including how dealt with difficult situations and how you make sure projects come to fruition. You can be asked very general job questions like "tell me about a time you dealt with a difficult coworker" to more data-science-specific queries like "how do you deal with a data science project where the model fails?" These questions won't necessarily have right or wrong answers; they'll instead be open to interpretation by the interviewer.

TIP Before the day of your onsite interview you can ask the recruiter or hiring manager about what the interview will be like. At a minimum they should give you a schedule of who you are talking to and what they'll be covering. If you ask for it, they may be able to give you some more specifics around what to expect during the technical and behavioral parts of the interview. This information will help you arrive prepared.

Going through an on-site interview is emotionally taxing. You'll have to quickly switch from thinking about technical topics to questions about yourself and your future dreams, all while presenting yourself in a professional and friendly way. Depending on the size of the company, there may be one or many people doing these interviews with you, and for each of these people you'll want to make a good impression. One of the best ways to manage nerves during the interview is to remember that the interviewers are people too, and they want you to perform well just as much as you do. They are your allies, not your adversary.

Let's dive deeper into the different parts of the interview. This section of the chapter goes into the different parts of the on-site interview, but check out the appendix for a detailed list of interview questions, example answers, and how to think about the questions.

7.4.1 The technical interview

For many data scientists, a technical interview is the most frightening part of the whole interview process. It's easy to imagine yourself stuck in front of a whiteboard being asked a question you have no idea how to answer and knowing you won't get the job. Just writing that previous sentence induced anxiety!

To best understand how to handle the technical interview, we need to reframe how to think about it. If you completed Chapter 4 of this book and have created a data science portfolio, you've already passed the technical interview. The point of the interview is to see if you have the skills necessary to be a data scientist, and by definition have those skills since you did data science! If during an interview you find yourself being judged for not being able to answer a tricky question, then that is a sign the interviewer is doing a bad job, not you. You have the necessary skills, you have experience under your belt, and this part of the interview is for you express that. If the interview doesn't allow you to do that it's not your fault.

In this process you are going to try and show the interviewer that *you have the skills needed for the job.* Showing you have a set of skills is a very different activity than answering perfectly every question you are asked. A person can give exactly the answers an interviewer wants and still do poorly in the interview, or they can give incorrect answers and do well. Consider two answers to a question in an interview:

> **Interviewer:** What is *k*-fold cross validation?
>
> **Answer A:** You randomly partition the data into *k* even groups and use them as test data for *k* models.
>
> **Answer B:** You randomly draw a sample of data and use it as test data for a model, *k*-times. Then you take the average of the models and use that. This is a really great method for handling overfitting since you have a bunch of models that all have different training data. I used this method on the main project in my portfolio where I predicted house prices.

Technically, Answer A is correct while Answer B is not (that's cross validation, but not technically *k*-fold since the data wasn't split into even groups). That being said, Answer A gave the interviewer no information except that the interviewee knew the definition, while Answer B showed the candidate knew the term, knew why it was used, and had practical experience with it. This illustrates why is it so important that during an interview you think about how to convey that you have the skills.

Specifically, there are a few things you can do in your data science technical interview that can help convey you have these skills:

- **Explain your thinking.** As much as possible, don't just give an answer, but explain why you got the answer you did. Giving an explanation provides the interviewer with

how you think about the subject and can show you are on the right track even if you didn't get the right answer. One word of caution: while repeating the question aloud might feel helpful (ex: "hmm, would a linear regression work here?"), there are interviewers who may take this as a sign you don't know much. Practice answering questions directly and practice how you frame your thought process from the start.

- **Reference past experiences.** By talking about previous projects or work you've done, you are repeatedly grounding the conversation in your real-world practical skills. This can make an ambiguous answer more credible and concrete, or just give an alternative topic of conversation if your answer is a little off-mark. You have to do this in moderation—if you spend too much time talking about your past instead of talking about the question at hand it may seem like your avoiding it.
- **Be open and honest if you don't know the answer.** It's totally possible (and normal!) to not know the answer to every question in the interview. Try to be upfront about it and explain what you do know about the answer. For instance, if you were asked "what is a semi-join" and you don't know the answer, you could say something like, "I haven't heard of that kind of join before, but I suspect it might be related to an inner-join." Being open about what you don't know is better than confidently being incorrect—interviewers are often wary of people who don't know what they don't know.

TIP when answering interview questions your instinct might be to answer as quickly as possible. Try to fight that instinct—it's much better to wait a few questions and give a strong answer than quickly say a weak one. With the stress of an interview it's hard to slow down your speaking, so practice answering questions beforehand to get more comfortable.

Here are the general types of questions you'll see in the technical interview. Again, check out the appendix for a set of examples of these.

- Math and statistics – these questions test how well you understand the academic topics that are a necessary foundation for a data science job. They include:
    - Machine learning – this includes knowledge of different machine learning algorithms (*k*-means, linear regression, random forest, principal components analysis, support vector machines), different methods for using machine learning algorithms (cross validation, boosting), and general expertise on using them in practice (like when certain algorithms tend to fail).
    - Statistics – you may be asked purely statistical questions, especially if your work is in an area that would need it, like experimentation. These questions can include statistical tests (like *t*-tests), definitions of terms (ex: ANOVA and *p*-value), and questions around probability distributions (like finding the expected value of an exponential random variable).
    - Combinatorics – the field of mathematics that covers all things related to counting. These are logical problems like "if a bag has six different colored marbles in it, how many combinations are there if you pull two out without replacement?" These

questions have little to do with the job of a data scientist, but interviewers sometimes believe that they give insight into your problem-solving skills.

- Databases and programming – these are questions to test how effective you would be at the computer-based parts of a data science job. They include:
  - SQL – in almost any data science interview you will be asked questions about querying databases in SQL. This knowledge is needed for most jobs and being familiar with SQL indicates you should be able to quickly get started in your new role. Expect to be asked questions on how to write SQL queries for sample data. For example, you may be given a table of student grades in multiple classes and asked to find the names of the best scoring students in each class.
  - R/Python – depending on the company you may either be asked to answer general programming questions by writing pseudo-code or to solve particular questions using R or Python (whichever language the company uses). Don't worry if you know R and the company uses Python (or the reverse); usually companies are willing to hire and train in the new language on the job since there is a lot of transferable knowledge. Expect to have a question where you have to actually write code (ex: how would you filter a table in R/Python to only include the rows above the 75th percentile of a score column?).
- Business domain expertise – these questions will be highly dependent on the company you're applying to. They'll be used to see how much familiarity you have with the type of work the company does—while this knowledge is something you can pick up on the job, the company would rather you had it than didn't. Here are some sample questions from different industries:
  - Ecommerce company – What is the click-through rate of an email? How does it compare to open-rate, and how should they be thought of differently?
  - Logistics – How do you optimize production queues? What are some things to think about when running a factory?
  - Non-profit – How should a non-profit try and measure donor growth? How could you tell if the too many donors are not renewing?
- Tricky logic problems - outside of problems that are related to things having to do with data science, you may get general "brain-teaser" questions in your interview. These questions have the reported purpose of trying to test your intelligence and ability to think on your feat. In practice, the questions don't do anything of the sort. Google did a massive study and found these sorts of questions had no ability to predict how a candidate would do on the job, and only served to make the interviewer feel smart. These tend to be questions like "how many shampoo bottles are there in all the hotels in the US?" For larger companies you can often Google to see if they use these types of questions (and which ones).

It's hard to say exactly which of these questions you'll see and how much time will be spent on each of them—it highly depends on the company and the person interviewing you.

Try your best to remain calm and confident as you work through them, even if you are unable to answer some of them. If the interviewer is talking to you as you give a partial answer, they may be thinking that you're doing well and be willing to point you in the right direction. Often the questions are designed to cover so many different topics that no data scientist could answer all of them, so by design you'll have some you won't be able to get.

Interviewing the interviewer

Each time you meet with a new person during the onsite interview, they will end their part with "do you have any questions for me?" This is one of your only opportunities to get candid information about the job, so use the time wisely! Here you can find out more on the technology used, the work, and how the team functions. As discussed earlier in the chapter, it also shows that you have a sincere interest in the company, so it's worth thinking of questions in advance. Here are some example questions:

"What technologies do you use, and how to your train new hires on them?" This question is great if they haven't gone into the detail about the technology stack, and it will give you insight into if they have formal training processes or just hope employees pick up the knowledge on their own.

"Who is our team's stakeholder, and what's the relationship with them like?" This is you asking who is going to be calling the shots on the work. If the relationship is poor, you may end up being a slave to whatever the stakeholder's demands are, regardless of if that goes against your judgement.

"How do you do quality control on data science work?" Since the team doesn't want to put out work with errors, there should be some checks in their process. In practice, many data science teams have no checks, and blame the creator of the work when things go wrong. That's a toxic workplace to avoid!

7.4.2 The behavioral interview

The behavioral interview is designed to test your interpersonal skills and give the data scientists on the team a better understanding of who you are and what your background is. While technical questions will all be bunched into one or a couple blocks of time, the behavioral questions may appear throughout the whole on-site interview. They can happen in a one-hour chunk with an HR representative, a ten-minute window as closing questions from a technical interviewer, or even as small talk with one employee while you wait for another to arrive. So be ready to answer behavioral questions at any time.

Here are a few examples of interview questions you should expect to receive:

- "Tell me about yourself" – This question showed up in the phone screening and can show up every time you start talking to a new person. Again, try and give a 1-2 minute summary, but this time cater it to the person you're talking to.
- "Describe a project you've worked on and what you learned from it." – This question is intended to see if you can look at a project in your history and be introspective about it. Have you processed what went well and what didn't?
- "What is your greatest weakness?" – This is an infuriating question because it *seems* like from a game theoretic perspective you want to answer with something that shows as little weakness as possible. In practice, what they are trying to do is check that you

understand your own limitations and have areas where you are actively trying to improve.

Notice that the questions are all very open ended and have no right or wrong answer, however there are ways of expressing yourself that can dramatically improve how your answers are received.

For most of these questions, especially ones asking about past experiences, your answers should follow a general framework. (1) Explain the question in your own words to show that you understood it. (2) Give an explanation of a past experience where that situation occurred, focusing on why the problem existed. (3) Describe what action you took to resolve the problem, and what the result was. (4) Give a summary of what you learned from it.

For example, let's take the question "tell me about a time you delivered something to a stakeholder and you got a negative result." A response could be: "So you're asking about a time I disappointed someone with my work [1. explaining the question back]. That happened on my last job, where I had to do a report on customer growth. Our team was juggling a lot of different requests, so I didn't have much time to focus on a request from one director. When I handed over a report I only spent a day on, she was very disappointed with it [2. describing the problem]. I first apologized for not meeting her expectations, then worked with her to see how we could reduce the scope of the request and still meet her needs [3. providing a solution]. From this I learned it's best to let someone know early if you can't meet their request so you can come to a mutually agreeable solution [4. what you learned].

A nice thing about behavioral interview questions is that they are all similar enough to each other that you can plan answers in advance! If you have three or four stories that cover working in a challenging situation, handling a difficult teammates, and managing failure, then you can draw from those stories for most interview questions. This is much less stressful than trying to think up a past story and improvise on the fly. If you have the time, you can practice telling this career tales out loud by yourself to a friend to learn how to best structure them.

An interviewer asking you to describe a past project shows up in almost every interview, so it's worth being well-prepared for that. The ideal project for a story covers the points listed above: it was a challenging situation where you overcame adversity, especially difficult teammates, and then you found a solution. The story should ideally fit into the four-step answer style listed above too. That being said, it's often difficult to find a story of a project that has every interesting twist and turn in it, especially if you're an aspiring data scientist. If you find yourself lacking in an answer for this, try telling a simple story of a project in your portfolio from Chapter 4. Even an answer of "I thought it would be interesting to analyze an unusual data set, so I obtained it, cleaned it, and found an interesting result I blogged about" shows the interviewer that you can work through the task of doing an analysis.

There is plenty more on this topic of answering these behavioral questions, such as thinking about the wording and how long you should spend on them. That being said, most of the techniques aren't data science specific, so you can easily go deeper by reading general

interviewing books and articles, which we give recommendations for in the Part 2 resources section.

7.5 Step three: the case study

If you did well on the on-site interview you'll be asked to complete a case study: a small project that will show the company how well you do data science in practice. Someone on the data science team will you give a data set, a vague problem to solve with it, and a set time period to do it in. This might either be during the on-site interview with an hour or two to work, or a longer period like a weekend so you can do it at home. You'll generally be allowed to use the programming languages or tools you are most familiar with, although it's possible the company will limit you to just the tools they use. After your time has elapsed, you'll share your results in a short presentation or discussion with a group of people from the data science team. Some example case studies are:

- Given data on promotional emails sent by a company and data on orders placed, determine which of the email campaigns did best and how the company should market differently in the future.
- Given the text of 20,000 tweets where the company was mentioned, group the tweets into different topics that you think would be useful to the marketing team.
- An expensive A/B test was run on the company website, but halfway through the data stopped being consistently collected. Take the experiment data and see if there is any value that can be derived from it.

Notice that in each example case study, the objective isn't directly a data-science question. Questions like "which campaign did best?" make sense from a business context, but there isn't a "which campaign did best?" algorithm you can just apply! That is why these case studies are so useful as interviewing tools: they require you to go from the very beginning of a problem all the way to a solution.

With that said, what exactly is the company looking to see out of a good case study? They want to know:

- Can you take a vague, open-ended problem and figure out some methods to try to solve it? It's entirely possible you won't solve the problem, but as long as you make an attempt in a reasonable direction, it shows you have the technical skills and can get things done.
- Can you work with real-world messy data? The data you'll be given will likely require filtering, joins, feature engineering, and handling missing elements. By giving you a complex dataset, you'll be doing the kind of work that you would do on the job.
- Can you structure an analysis? Will you look at the data in a methodical, well-thought out way, or will you investigate things that don't relate to the task at hand?
- Can you produce a useful report? You will have to create a presentation about your work, and possibly documents like Jupyter notebooks or R markdown reports. Can you

make something that is useful to the company? Can you structure a useful narrative?

The good news is that the skills and techniques that are needed for great case studies are the exact same ones that you need for good portfolio projects: taking data and a vague question and producing an output. Even better if you made a blog post—that mimics creating a presentation for an interview case study!

There are a few minor differences between a portfolio project and a case study problem that you will want to take into account. First, in a case study you have a limited amount of time to do the analysis. This can be set by the calendar, as in one week from the day you get the materials, or set by the hours worked, as in no more than 12 hours of time spent. That short amount of time means you'll want to be strategic on where you spend your hours. In general, the data cleaning and preparation steps take much longer than data scientists expect. Just getting tables to join together, filtering malformed characters out of strings, and loading the data into an IDE can all take a lot of time and typically that part of the work won't look impressive to the company. Try not to get overly focused on preparing the data as best as possible if it means having too little time to do an analysis.

Another difference with case studies is you are judged on your presentation of results. This means you want to have a well-polished presentation with really interesting results in it. However, the act of creating a presentation can feel both uninteresting and less important than doing the analysis itself, so many data scientists put creating the presentation off until the very end. Putting the presentation off until the end is bad because you may run out of time, or you might find out that the analysis isn't as interesting as you hoped and you don't have time to make changes. As much as possible, start working on the presentation early and build it as you progress in your analysis.

One final difference with a case study is that you have an extremely specific audience: the small number of people you are presenting to. With a portfolio project you don't really know who will look at it, but with a case study you can hyper target your analysis. If possible when you are first given the case study, ask who the audience of the case study presentation will be. If it's all data scientists, then you can make your presentation more technical, like including details of the machine learning methods you used and why you chose them. If it's all business stakeholders, then try to be light on the technical components and focus more heavily on how your findings would impact the business decisions. If it's a mix of data scientists and business stakeholders, then try to include enough of each that if one of the group dominates the discussion you have enough to placate them.

The presentation itself is usually around 20-30 minutes of you presenting your findings with 10-15 minutes of questions from the audience on your approach and findings. It's a good idea to practice your presentation in advance and plan for what you want to say during each part of it. Practicing in advance also helps you keep to the allotted time—you don't want to talk for only 5 minutes or for 50 minutes straight. During the question and answer section you'll be peppered with questions on all sorts of topics. You may go from answering a question about a parameter in your model straight to a question about the business impact of what you

found. A good practice is to take a moment to think about the question before answering so that you can get your bearings and think through your response. In the event that you don't have an answer you are confident in, it's generally best to give some version of "I am not sure, but…" and give some ideas on how you could potentially find the answer. If possible, add the relevant context you do know.

7.6 Step four: the final interview

After the case study is over, and likely during the same trip to the office as your case study, you'll have one last interview. This interview will be with a person who makes the final call, such as the manager of the data science team or the director of engineering. Depending on how the company runs the interview process, this person may have already been prepared with information on how you did in the earlier parts of the process, or they may know literally nothing about it. The objective of this interview is for that final person to give the green light on you being hired.

It's hard to know exactly what questions you will be asked during the final interview because it depends so heavily on the person interviewing you. A more technical person may focus on your technical experience and what skills you have, whereas a business person may ask you more about your problem-solving approaches. Regardless of the type of person doing the interviewing you should definitely expect questions that are the same style as those in the behavioral interview: "how do you handle difficult situations and deal with problems?" For these questions being open, honest, and sincere will go a long way.

Following up

After each part of the interview process, you may be inclined to contact people from the company in one form or another. Following up can be used to show gratitude to the people you've met with, as well as gain you more information on how the process is unfolding. However, if done poorly, reaching out can come off as abrasive or desperate and jeopardize your chances of getting the job. There are really three different ways you can follow up, depending where you are in the process.

Before anyone from the company has contacted you: If you have sent your application but haven't heard back do not follow up—them not responding is a sign they are not interested.

After contact but before you've met anyone in person: For example, after having done the phone interview, you should only follow up if you are unsure of your status in the process. You can follow up with one email only if it's past the timeline for when they said the next step would happen. In that case simply ask for a status update.

After in person contact: You can, but in no way need to or necessarily should, email the people who interviewed you a brief thank you. If you don't hear back past when they said you would, you can also email the recruiter asking for an update.

7.7 The offer

If all goes well within a week or two of your final interview you'll get a call from someone at the company letting you know that they are going to extend you an offer. Congratulations! In the next chapter, we'll go into much more detail on what makes a good offer, how to compare different offers for different data science jobs, and how to ask a company to improve what they are offering.

Unfortunately, there is also the possibility of the disappointing case that you don't get an offer from the company. After taking a moment to grieve the loss of the potential job, you can view this as an opportunity to learn what areas you can improve for your next interview. If you only made it to the initial phone screening, that's a probably a sign that your base qualifications weren't a fit for the particular role. In that case, you should consider adjusting what job postings you are apply to. If you made it to the on-site interview or case study but no further, there is likely a specific reason that you weren't a fit for the company or role—try and deduce if there is something you could focus on for the next interview. If you made it through the final interviews but didn't get the job, that usually means you were well suited for the position, but someone else happened to be a slightly better fit. In this case there isn't much to do but continue to apply to similar roles. You shouldn't reach out to the company asking why you weren't hired—you're unlikely to get an honest answer and it's viewed as unprofessional.

7.8 Interview with Ryan Williams, Senior Decision Scientist at Starbucks

Ryan Williams recently transitioned from being a manager of data science at a marketing and sales consulting firm, where he ran the data science interview process. Now at Starbucks, he helps inform decision-making for the Starbucks Rewards program as part of their analytics & insights team. He has a bachelor's degree from the University of Washington where he dual majored in statistics and economics. Prior to joining Starbucks his career was focused on the consulting industry.

7.8.1 What are the things you need to do knock an interview out of the park?

The big thing in general is just preparation. There's a whole skill set that goes into interviewing that is very specific to interviewing. A lot of people think they can just walk into an interview and their experience is naturally going to shine through. I have felt that myself, and I've seen other people come into interviews with that mindset. But with questions like "tell us about a time you had trouble communicating," unless you've really prepared you can babble and talk in circles. There's really a skill set that you need in interviewing, and the way to prepare for that is to read up on the typical questions you're going to face. No matter where you're interviewing, you're going to face some behavioral questions, you're going to face some technical questions, and you're going to face some business case questions.

So know the common behavioral questions. Know the types of technical questions people are going to ask and understand the business case type questions. Unless you're prepared in the right ways to demonstrate your experience, you're not necessarily going to get that opportunity in an interview.

7.8.2 How do you handle the times where you don't know the answer?

One case is that I remember I was interviewing at one of the bigger tech companies and they asked lots of harder and harder stats questions. It got to the point where they asked me something that was super academic. I think it was that given a certain probability distribution function, how would you use its moment generating function to find the kurtosis of the distribution. I was like well... that's something maybe I could have answered in college, but I definitely can't now.

That being said, when I gave my answer, the interviewer was clearly disappointed. I could go on a whole rant about it, but I wasn't happy because I feel like that was the type of question that's really more trivia than something that demonstrates my thinking. Having a job is about resourcefulness in your ability to solve things you don't know. It's not about your ability to come into a room knowing everything you need to know already.

7.8.3 What should you do if you get a negative response to your answer?

There's the emotional component where you're going to be unhappy with the fact that you were unable to answer the question, but you have to not let that tank the interview for you. It's natural to keep thinking about what you could have done to get that problem right instead of what you could do to answer the next question. But you really have to just be able to move on and stay sharp about the questions that you're going to get in the future.

I would say if you are running into the questions like that, you need to be interviewing the company too. They're asking you a lot of questions, but you should be using the types of questions they're asking you to infer whether this is actually a company you want to work for. For me, a company that thinks that it's super important for a data scientist to be able to derive a moment generating function three times to get the kurtosis is not necessarily the type of environment that I feel is going to be best for me to work in.

7.8.4 What has running interviews taught you about being an interviewee?

I am a lot more thoughtful about the types of questions I'm being asked when I'm being interviewed and also the types of questions that I'm asking in an interview. Before I ever interviewed people myself, I took the interview questions at face value. Being interviewed felt like taking a test because the interviewer's asking a question and I thought I only needed to answer it right. I wasn't evaluating the types of questions being asked, which now I'm a lot more conscious of. Is this interviewer asking me questions that are a lot of trivia? Do they want me to whiteboard a lot of programming problems? Do they care about the things I care about in that data science role?

7.9 Summary

- The interviewing process is similar across companies for data science
- For on-site interviews, expect technical and behavioral questions
- Be prepared to do a data science case study at home
- Take time to prepare for and practice answers to common interview subjects
- Know what you want to learn about the company and role during your interview

8

The Offer: Knowing What to Accept

This chapter covers

- Handling the initial offer
- Negotiating your offer effectively
- Choosing between two "good" options

Congratulations! You've got a data science job offer. This is a big accomplishment, and you should definitely take a little time to savor it. You've done a lot of work over the past months or even years to get to this point.

This chapter will help you respond to and decide on the offers you receive. Even though you're probably really excited, you **should not** immediately say, "Yes! When can I start?" when you get an offer. All data science jobs are not created equal – you've been selective in where you've applied, but you may have found out things in the interview that raised concerns. Or you have a benefit that is a hard requirement for you, like good health insurance for your family, and you need to look over the offer details. Even if you are sure you want to take an offer, you should still not say yes immediately – you want to negotiate! Having an offer you haven't accepted yet is when you have the most power. Now that this employer has finally found someone they're excited about (you!), they want to close you. Recruiting is very expensive – it takes up a lot of time from HR and the data science team to evaluate applicants and interview, and every week without the new hire is a week where the company doesn't benefit from the work of that (hypothetical) person. Use this opportunity to ask for what's important to you, whether that's a higher salary, working from home once a week, or a higher budget for attending conferences.

8.1 The process

In general, the offer process goes something like this.

1. The company tells you an offer is coming—as soon as possible they want you to know about the offer so you don't accept an offer from a different company before theirs arrives.
2. They give you the offer—either by email or phone (then followed by an email), they let you know the details on salary, start date, and other necessary things for your decision. They usually also give you a date you have to accept or decline the offer.
3. You give them an initial response. As discussed in 8.2, unless you're absolutely sure you don't want to take it, we recommend expressing your enthusiasm and asking for a few days to a week to think about it, rather than immediately saying yes. When you have your next conversation with them, you'll begin the negotiation process (section 8.3).
4. They may be able to give you an answer immediately when you negotiate, but often they'll need some time to let you know whether or not they're to give you a better offer. In either case, you have to decide if this is good enough for you and return your final decision.

8.2 Receiving the offer

The call or email you receive with your offer is usually either with hiring manager or the recruiter or HR person you've been working with. In either case, your response should be the same.

Start by saying how happy and excited you are that you got an offer. If you sound unenthusiastic about working there, a company is going to get worried that, even if you take an offer, you won't stay there long and won't be contributing your best work.

> **WHEN YOU'RE VERY DISAPPOINTED** While we generally recommend waiting to start negotiating until you have all the details in writing and have had a few days, if you're very disappointed in the offer you'll probably want to start discussing that then. For example, let's say the salary was 25% less than you expected. You could start by saying, "Thank you so much. I'm really excited about this opportunity and the work I'd be doing at company Z. But I want to be honest that the salary is a fair amount lower than I was expecting. I know that the market rate for someone like myself with a master's and five years of experience in New York City is in the range of X to Y. What can we do to get the offer more aligned with that range?" You could also use a higher competing offer or a higher current salary as reason for asking for more. Reserve this tactic for if the salary is something that would make you immediately turn down the job, not one where you'd take it but want to negotiate 5% more. Getting 20% more may simply be impossible, no matter how good a negotiator or candidate you are, and it's better to find that out earlier than later

They should tell you that you'll be getting the details in an email and if not you should request it. This serves two purposes. One, you get to sit down and take time to read through

it all and consider the whole package, without frantically taking notes during a phone call and trying to decipher your handwriting later. Two, you should never consider a part of the offer "official" until it's in writing. While most of the time it won't be a problem, you don't want to get into a situation where there was a misunderstanding over the phone and you think you accepted a certain salary and benefits and it turns out that's not the case. When you receive the offer, it should include your title, salary number, any options or stock units offered, and the benefits package. If it doesn't include the details you need, like the full explanation of health insurance benefits, you can ask to get that as well.

Finally, they should give you a time-window to accept the offer. The time window should be at least a week, and if it's less than that ask for it. The best thing to do is be confident and simply say "I need a few days to consider this." If you find yourself unable to bring yourself to be that solid in your convictions then this is a great time to invoke someone else, like a partner, family member, or lucky goldfish, who you need to discuss it with. It helps to have an external decision-maker and constraint. That way, the recruiter or manager knows they can't just pressure you on the spot.

Sometimes companies will give you an "exploding" offer, where you have to respond in less than a week or the offer will be rescinded. It should never be as short as needing to respond in that initial phone call, but it might just be 24 hours. Usually there is room to push back on this by saying something like, "I know we both want this to be a great fit. Choosing my next company is a big decision, and I need a week to carefully consider it." In the worst-case scenario a company will refuse, for some reason, to wait for you to have the time to fully evaluate of the offer. If you find yourself in this situation it's a huge red flag. A company that isn't willing to respect your needs here is a company that won't respect your needs in other places. By giving you a small window, they know they're pressuring you to act quickly, which causes more anxiety and maybe make a worse decision. Despite many people applying to data science jobs, companies are still struggling to find people who are good fits, so a company that would treat people that way has something strange happening in it.

If you're interviewing with other companies, let those companies know you received an offer and when you need to respond to it by. If you're in the final rounds, they may be able to speed the process up so they can get you an offer before you need to decide on your current one. It's totally normal to update other companies when you receive another offer and they will appreciate it. When you reach out to them, reiterate how excited you are about their company and the work you'd do there. Some companies may not be able to do anything, but at least you give it a chance. Again, since companies are often struggling to find data scientists to hire, if they have someone they like in the interview process, they generally can move quickly to speed things up.

8.3 Negotiation

A lot of people hate negotiating a job offer. One reason is that it's perceived as a strictly zero-sum game: if you gain, they lose. Another is that it feels selfish and greedy, especially if the

offer is better than your current job. It's easy to look at a situation and not feel like you deserve more money than a company offers, and that you're lucky to have any offer at all. But when you are in this position, you owe it to yourself to do your best to maximize your offer.

Because you, yes you, are the person best suited for this job. We all face impostor syndrome, but the company sees in you what they want for this job. As mentioned in the beginning, at this moment you're in your strongest position to negotiate. And companies expect you to! You want to be prepared. An appropriate salary for you isn't related to what you made before or what the company may initially offer, but instead what all the other companies are offering for people with your skillset. Make sure that you get paid as much as your peers, that the total compensation matches your expectations, and that you get the benefits that are most important to you. You have a chance to make 5% of your salary in a five-minute phone call - you can handle the discomfort for that long.

The need for negotiating is especially true in data science, where there is a massive disparity in what people are making. Because of how new the field is, and how many different roles fall in data science, there aren't clear standards in what people make. Two people can have wildly different salaries with the same skillset, just by one person calling themselves a data scientist and another a machine learning engineer. These differences in salary get compounded as people move to new, higher paying jobs. At this point there is little correlation between the amount a person is making and the qualifications and abilities of the employee.

8.3.1 What is negotiable

The first thing many people think of when negotiating a job offer is salary. Before you reach the final rounds of interviews, you should research the salaries of not just data scientists in general but for that industry, the city the company is located in, and the particular company if they already have data scientists. While the offer may be a big salary jump, remember that you want to be paid similarly to your peers at your new company. You are not obligated to let the company know your current salary during the interview process, and in some cities and states it is illegal for companies to ask. If you go on Glassdoor and find the average data scientist makes twice as much as you do, that's great.

During the interview process if you're asked what you currently make or what your salary expectations are, try to avoid giving an answer as you may risk they'll end up offering you less because they'll know it will meet your expectations. You can try answering, "Right now I'm more focused on finding a position that will be a good fit with my skills and experience. I'm sure if we find a mutual fit we can come to an agreement on an overall compensation package." If they keep pressing you, such as by saying they need to know to make sure you won't expect something out of there range, you can reply, "I understand you want to make sure my expectations match your compensation band. What is your range for the position?" You could also say, "I would need to ask a few more questions to the [hiring manager/senior data scientist/any person you'll have in the interview flow who you're not talking to right now] get a better idea of what the position entails before I can provide a realistic expectation." If they refuse to give a range and won't move forward if you don't give a number, that's a bad

sign. If you absolutely need to give an answer because you love the company and you've tried deflecting multiple time, say something like, "While I am certainly flexible depending on the overall package, I know from my research that $X to $Y is standard for someone with my experience and education."

On feeling imposter syndrome by Caitlin Hudon

Making big career changes is a time when many people—even experienced, senior data scientists!—feel pangs of imposter syndrome. Imposter syndrome is the feeling of doubt over your accomplishments and worrying about being exposed as a "fraud." Salary negotiation in particular is tricky: it's a high-stakes situation and a literal exercise in determining your worth. Your feelings about your worth will almost certainly play into your decision-making, so it's important to combat feelings of imposter syndrome so that you can be objective and assertive about all that you bring to the table.

Imposter syndrome is especially common in data science. Depending on who you ask, a data scientist is some combination of an analyst / statistician / engineer / machine learning pro / visualizer / database specialist / business expert, each of which is a deep position in its own right. It's also a constantly expanding field where there can be a lot of pressure to stay "on top" of new technologies. So, when we boil it down, we have people from a variety of backgrounds coming to a new field with many applications whose boundaries aren't clearly defined (thus causing inevitable gaps in their knowledge of that field as a whole), and where technology is changing faster than a single person can keep up with. If this feels like a lot for a single person to contend with, that's because it is!

This brings us to my approach to combating imposter syndrome, which is to focus on my unique experience and how it makes me unique, rather than to compare myself to an impossible-to-achieve data scientist ideal. To get there, I've accepted that I will never be able to learn everything there is to know in data science—I will never know every algorithm, every technology, every cool package, or even every language—and that's okay. (The great thing about being in such a diverse, growing field is that nobody will know all of these things—and that's okay too!)

Furthermore, you and I know things that others don't, and have experience that others do not have. The knowledge and experience you have both overlaps with others and also sets you apart from others, but is not a subset of others. Focus on your own unique experience. You've had to earn it, after all, and it's important to remember that it sets you apart from other data scientists.

When you're in the midst of salary negotiations, if you find yourself up against your own imposter syndrome, take a minute to think about all of the skills that you've learned, the problems you've solved, and all of the great assets you'd be bringing to your future team. Your experience is valuable and you should be paid fairly for it. Don't be afraid to ask for what you deserve.

You should not feel bad for being paid what you think is reasonable. What you think is reasonable shouldn't be defined by what you have made before, but only by what you want to make in the new position. Companies frequently offer a lower amount anticipating you will negotiate up to what they expect to pay a new hire, so it's very common you can get at least a 5% increase

But there's much more beyond salary you can negotiate for. First, there are other direct monetary benefits, like a signing bonus, moving allowance, and stock. A signing bonus is easier for the company to give than the same amount of salary increase, as it's only a one-time thing. The same goes for a moving allowance - if you'll be moving, ask about the company's relocation policy. If it's a big company, there's likely something standard and they may even work with a particular moving company to help. Even small companies may be able to give you something; you won't necessarily find out unless you ask.

But you can go broader even than that. Go back to what you thought about in the job search process – what's important to you? Some things are hard to negotiate – the healthcare options and 401k match for example are often set by HR and standard for everyone in a role. These are things that you want to consider earlier in the process – for non-startups, you can usually find employee reviews on Glassdoor with information about the benefits package. But there are lots of things you can ask for now, like:

- a flexible or remote work schedule
- an earlier review (six months vs. a year) which could get you a raise faster
- educational benefits
- a budget to go to conferences

When doing this, keep in mind the restrictions of the organization you are joining. Non-profits are unlikely to have much room to negotiate salary, but they might be flexible on hours or vacation. A company where the data team is already distributed is more likely to let you work from home a few days per month than if you join a team where everyone is in the same office. Ensure that what you're asking for is in writing and honored—you don't want to negotiate an early review and then when you arrive on the job never get it. Also keep in mind these sorts of non-salary negotiations likely will only make adjustments on the margins—if the salary is far below what you want, there is no way these changes could make the offer acceptable.

8.3.2 How much you can negotiate

Your best negotiation lever is a competing offer. A competing offer lets a company know that there is a market for your services at a higher price and you have another place to go. There are some big companies that even will only negotiate if you have a competing offer. If you find yourself with two offers, it's best to make statements like "I would much rather work with you, but company ABC is offering me this much more; can you match that?" Don't lie; if you try to say that to both companies it could easily come back to haunt you.

Another piece of leverage you have is your current job – most companies know you won't want to take a pay cut. An exception might be if you're breaking into the field from something completely unrelated and have to move down some ranks in a company hierarchy. But if you're relatively happy, or have better benefits at your company, use that. Even if they may not be able to match exactly, you could get an equivalent bump in a different benefit or salary. For example, let's say your current employer offers a 3% 401k match and the offering

company offers none. That match effectively raises your salary by 3% vs what the new company is offering. This can be especially helpful if you're transitioning to data science from a lower paying field, so your salary will increase a lot, but you'll be losing out a benefit you get at your current job.

When placing a value on your skills, consider how uniquely your background suits your new job. You don't necessarily have to be an experienced data scientist. For example, it would be great if you were one of the three people to earn their PhD in AI at Stanford this year. But let's say your new job is as a data scientist helping sales and you used to be in sales. Having that domain knowledge is a huge advantage that even more experienced data scientists might not be able to bring to the position. The more it sounded like the position was designed for you when you saw the job posting, the more leverage you are likely to have.

Sometimes it can be difficult to recognize the full picture of a job offer. If you'd need to move, for instance, you should consider moving costs and also changes in the cost of living. A $90,000 salary in Houston will go a lot farther than a $95,000 salary in New York City. Tally up all the extra benefits. Things like a good healthcare plan or 401k match can be worth thousands or tens of thousands of dollars. In general, don't lose the forest for the salary tree.

If you've negotiated and you get everything you asked for, they expect you to accept! You should only start the negotiating process if you'd take the offer if they met all of your conditions. Otherwise, why bother? One reason could be you have another offer you really want to take, and the better this one the more you can negotiate the other one. This is an extremely risky tactic, and you want to be careful about possibly burning bridges. Even if you may never work at the company that you use as leverage, the employees who interviewed you or made your offer may remember.

It is very, very rare for a company to pull an offer because you negotiated. If they did, that's a sign you definitely do not want to work there. It is totally normal to respectfully negotiate and rejecting a candidate for that is a huge red flag. Take it as a bad situation avoided.

A brief primer on RSUs, options, and employee stock purchase plans

This is intentionally called a "brief primer," not an "exhaustive overview." We highly recommend researching more on whatever is contained in your offer letter.

Generally, all of these will vest over a four-year period with a one-year cliff. The "one-year cliff" means if you leave before one year, you get nothing. Instead, you get the first year's worth all at once at your one-year anniversary. After that, they will generally vest every quarter for the following three years.

RSUs: These are restricted stock units. You'll see a dollar amount given to you an offer. If you take the price of the stock at the time of your offer and divide the monetary amount by it, you'll get the number of shares per year. That means if the shares go up in value, your compensation goes up. When they vest, you'll get them as stock at that time – your company will usually hold back the amount of tax in the form of shares.

For example, let's say you get an offer of $40,000, vesting over four years with a one-year cliff. Right now, the stock is trading at $100 a share, so you'll get 100 shares after one year, then 25 each quarter afterward. After working there for

one year, you'll get 65 shares (35 held back for tax). At that point, you can do whatever you want with them, complying with the company rules (usually you can only sell stock in certain time periods). If you're at a private company, the vesting will work the same, but you won't be able to sell them.

Stock options: at a private company, there is no publicly traded stock available, so you'll generally get options instead. Options gives you the "option" to buy a certain amount of stock at a certain price. If the stock is trading way above that price, that's great news – if you can buy some for $10 and it's trading at $30, you'll make an instant $20 if you just buy them and sell them right away! If the stock does not trade above that value though, they're worthless – you don't want to buy a stock for $10 if you could buy it for $5. These are really great for early employees at a company that gets big. But they can be like lottery tickets. As long as your company stays private, the options have limited value, and even if the company goes public, the stock may end up trading below your option value. Unlike RSUs, which are generally going to be worth something, options may never be worth any money.

Employee stock purchase plans: this allows you to buy company stock at a discounted price. You contribute to the plan through a payroll deduction and then reach a purchase date, where your money is used to buy shares of the company. These offer two benefits. One, there is the discount on stock price, which may be as much as 15%. The other is a "lookback provision" – if the price has gone up between the beginning of the offering period and the purchase date, you'll get to pay the lower previous price.

For example, let's say at the beginning of the offering period the company stock is worth $10. You contribute $9000 over a year and then reach the purchase date. At a discount rate of 10%, you get to buy 1000 shares ($9000 / $9). If the stock price is now $20, those 1000 shares are worth $20,000, meaning you could sell them and get $11,000 in profit!

8.4 Negotiation Tactics

Now that we've gone over the big picture, let's walk through some specific negotiation tips:

1) **Remember to start by showing you're thankful and excited** – hopefully you are both! You want them to feel like you are on their side and you and the organization are working together to get a solution. If you don't feel like you're working together that's a red flag against working there.

2) **Be prepared** – before the phone call, prepare notes about what exactly you are looking to change in the offer and what total compensation you would like. During the heat of the moment it's extremely easy to blurt out a lower number than you wanted to make a person happy in that moment. Having notes helps you avoid this.

3) **Listen to what the person on the other end is telling you**- if they stress that they don't negotiate salary but compensation is important to you, ask about more options or a signing bonus. Try and work with them to find a solution rather than trying to be immobile in your position. While negotiating can feel like a zero-sum game, it doesn't have to be! Something very important to you may not be as important to them for example, so it's easy for them to give. And they want you to be set up for success and happy in your role.

4) **Don't seem like you're only focused on money** (even if you are). This can come off badly. You want to show you're motivated by the work you'll be doing, the mission of the company, and your new colleagues. Seeming like you're only in it for the money can leave a bad taste in their mouth, and they're more likely to worry you'll jump at the chance to leave for a higher salary.

5) **Try to keep a communal focus**. For example, instead of "I really need more options," try, "I'm really excited about the long-terms growth prospects of your company and how I can help you succeed. That's why I'm interested in being even more invested in them by having some additional options."

6) **Package together your desires instead of going point by point**. That way, they'll get the full picture of what you want instead of feeling like every time they settle an issue there might turn out to be another one. That being said, you can hold back the substitutes you're willing to accept: for example, you can ask for a higher salary, more options, and to work from home one day a week, but if they can't do the salary, you can then ask about a signing bonus instead. If possible, try to list the importance of each too.

7) **Avoid self-deprecation**. You read how great you are above? Read it again. Don't undercut yourself by saying something like "I know I haven't worked as a data scientist before" or "I know I don't have a PhD, but …" You have exactly what they're looking for or they wouldn't have made you an offer!

There's a lot more advice and research out there on how to negotiate effectively, and we've shared to some of our favorites in the appendix.

NEGOTIATING AS A WOMAN For a while, the common research theory was "women don't ask" – that one reason women are paid less than men is because women don't negotiate job offers or ask for raises. However, recent research has shown at least in some areas, women do negotiate as much, they're just less likely to be successful. Unfortunately, there is some bias here. Some of the tactics we discuss in this section, like having a communal focus (using "we" instead of "I") and offering reasons for your asks are especially beneficial to women.

8.5 How to pick between two "good" job offers

Receiving two (or more!) good offers is a great problem to have! But you still have to make a choice. Of course, having multiple offers won't always lead to this problem – sometimes you prefer one job so much more that it's an easy decision. But what do you do if that's not the case?

The first step is to go back to earlier in the chapter – negotiate! If you really like one more, but there's some deal-breaker for you, see if they can change that. Be honest with the company about what you're looking for. As discussed, that competing offer is great leverage to have and makes it more likely you'll get what you're looking for.

Next, you can always ask to meet again with the hiring manager or potential co-workers to gather more information. For example, maybe you hadn't asked about the average number of meetings a week, how many teams you'd be working with, how requests are prioritized, or anything else that would be helpful to assess if this is a place you'd enjoy working.

When you've looking at an offer, consider the long-term. You may have a short-term concern – if you have high student debt or are starting a family soon, maybe you want to just take the best paying offer so you can start paying that down or saving. But if you're fortunate to be able to think beyond the immediate financial concerns, you want to focus on maximizing long-term potential. What kind of work will you be doing at each one? If one job will give you training to be able to move up two steps at your next post and one will have you doing things you learned on your first day at bootcamp, you shouldn't jump at the latter just because they have a better vacation package. Remember that salaries in data science can really jump from one position to the next, even by as much as 30-40%. If your resume is a little sparse and this is a good chance to put a prestigious name on there, then you might take a job at a slightly lower salary for a year or two.

Finally, don't be afraid to let smaller factors influence you. Does one have a shorter commute? A more spacious office? If they both meet your minimum criteria and are similar on your most important deciding factors, you can start bringing in the smaller ones.

In situations like this it's often possible where multiple life choices seem reasonable. Do you take the exciting but risky new startup job, or stick in your reasonable government contractor one? Do you take the offer that has you managing people, or continue to work as an independent contributor?

Often there is no objective way to decide with route is better. You can't tell if the new job would be a good fit until you actually take the offer. You can't know if you'll like managing until you become a manager. This is an infuriating fact of life.

If you find yourself struggling with these decisions, it can sometimes help to reflect on the fact that you can only do the best with the information you have. You can't expect yourself to see the future. If you make a decision and ultimately are unhappy with the outcome you can move on from it. You can quit a job, move back after changing cities, or go back to being an independent contributor. Life is complicated and you can learn a lot even from paths that don't lead to the outcomes you want.

When you do make a decision, turn down any other offers with grace. One, it's the polite thing to do, but two, the data science world is small. You don't want to reject an offer by saying, "I can't believe you gave me that lowball offer, you're a terrible company and completely unethical," and then find the person you yelled at is the hiring manager for your dream job five years later.

8.6 Interview with Brooke Watson Madubuonwu, a Senior Data Scientist at the ACLU

Brooke Watson Madubuonwu is a senior data scientist at the ACLU, the American Civil Liberties Union, where she provides quantitative support for litigation and advocacy teams on issues related to civil rights. She has a master's in epidemiology and previously worked as a research scientist.

8.6.1 What should you consider besides salary when you're considering an offer?

It helps to know your priorities before going into that process. It's easy to compare salary, but you also want to compare the day to day or month to month experience of working at each place. I tend to think about the aspects of the job in three categories - lifestyle, learning, and values. Lifestyle is about the day-to-day way that the job interacts with other parts of your life: where will I live? Can I work remotely? Will I be working evenings and weekends? Will I get to travel? Will I *have* to travel? What healthcare, childcare, and retirement investment options are available to me? Do I have the flexibility I need to care for my family? That's the lifestyle bucket. There's also the learning bucket. Am I going to grow in this role? What systems are in place to ensure that I'm growing? Am I excited to learn from my boss and from the team? Finally, I think about the values and mission of the company or organization and the specific team. Does this organization and this team work toward something that aligns with my values? Does the team value inclusion? Is this a product or a team that I want to continue to build? Each of those three buckets can have different levels of importance at different times.

I've negotiated for a title change before in a previous role, and you can negotiate for things like education and conference resources, work from home days, travel support, or equity. Data science is a vast field that's always advancing, so I think it's important to protect time in your day to continue developing your skills.

8.6.2 What are some ways you prepare to negotiate?

In the past, I've been very uncomfortable talking about my own value and advocating for my own worth. That's something that I have been trying to unlearn throughout my career. What can be really helpful is having allies in your corner. Have a close friend practice the negotiation with you and have them take your part. Many people, myself included, find it much easier to advocate for our friends than for ourselves. Listening to the way someone else describes your strengths and your needs can be very motivating, and it really helps to put yourself in that mindset. Ask yourself, "How would I talk about someone I love, who I knew was a great fit for this role, and who I wanted to succeed?" That's how you should talk about yourself.

8.6.3 What do you do if you have one offer but are still waiting on another one?

If you've been through all, or most, of the interview process with a company you're waiting for, it usually doesn't hurt to mention to them that you have another offer. You can also usually buy more time with the offering employer by asking for a week or two to review the details and discuss the offer with your family. Negotiating further, asking detailed questions about benefits, and asking to speak team members about the work or culture are all actions you can take to serve the dual purpose of collecting more information about the role, and buying more time.

But if you're still very early in the interview process at another company, you may not be able to speed it up, and you have to compare the offer on the table to your current situation. If you have just graduated or are between jobs, you may not have the financial luxury of waiting. But if you are currently employed and still not interested by the offer on the table, it may be worth it to you to remain in your current role until you find a better fit. My first data science offer wasn't a great fit, and though I was very keen to transition into the field, it was worth waiting for a better match.

8.6.4 What's your final piece of advice for aspiring and junior data scientists?

I would advise aspiring and junior data scientists to keep an open mind about titles. I think the data scientist title has this allure for people, especially for those coming up in data science majors, which didn't exist when I was in college. They can feel if they don't get a job as a Data Scientist™ right when they graduate, they've failed. But the day to day functions of a data scientist might exist in a data analyst, a researcher, or one of many other kinds of roles, which can be a really great proving ground for you to hone your skills. I built my coding chops as a Research Assistant, and later as a Research Scientist, for years before anyone called me a "Data Scientist," and the work was just as engaging and interesting. Even data entry work as an undergrad has shaped my thinking about the way data collection decisions inform the analytical possibilities. No job is too small if you're willing to learn as you go.

8.7 Summary

- Don't immediately accept an offer: get the details in writing and ask for some time to look it over.
- Negotiate, negotiate, negotiate! You can ask for a higher salary or other monetary benefits, but don't forget about things like a flexible work schedule or a conference budget.
- When you're weighting two good offers, remember to consider the long-term potential at each, not just the initial salary.

Part 3: Settling into data science

While starting your first data science job is quite an accomplishment, it's only the beginning of your data science career. Working at a company as a data scientist is quite different from doing data science as a hobby or as part of a course. There are all sorts of concepts you might need to pick up, from corporate etiquette to the proper way to put code into production. The huge difference between the expectation of what the role is like and what the work truly is can be a shock. This part of the book aims to provide a comforting cushion to soften that jolt. By reading this part, you'll know what to expect in a data science job and be better prepared to excel in it.

Chapter 9 is all about the first months on the job, from the first few days where you may feel totally lost through settling in as you learn more about the role, your coworkers, and the data. Chapter 10 provides a guide to creating good analyses, a large part of most data science roles, by making and executing a plan for an analysis. Chapter 11 discusses taking machine learning models and putting them into production. It introduces concepts like unit testing which are essential for more engineering-based data science roles. Chapter 12 is a deep dive into the extremely relevant task of working with stakeholders, which is often the part of a data science job people struggle with the most.

9

The First Months on the Job

This chapter covers:

- What to expect in your first few weeks as a data scientist
- Becoming productive by building relationships and asking questions
- What to do if you're in a bad work environment

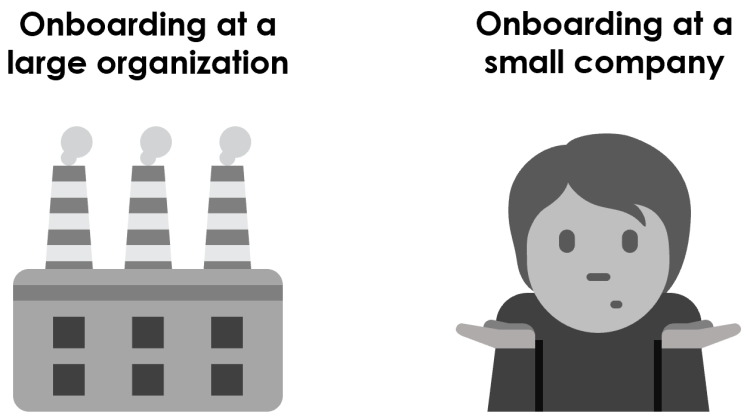
In this chapter, we're going to walk you through what to expect in your first few months and how to use them to set yourself up for success. These months will have an outsized impact on how the job goes-this is your chance to set up a system and support network that will allow you to be successful. While each data science job is different, there are some broad patterns as well as principles that can be applied to any job.

When you start working, you will instinctively want to get as much done as possible immediately. Fight that instinct. You need to be sure that you are not just accomplishing tasks but doing them in the right way. This is the easiest time to ask questions about how something should be done, since you aren't expected to know this at your new company. Managers occasionally forget that you don't have the institutional knowledge that your predecessor may have had, so you might get tasked with something that doesn't make sense to you. You might be able to fake your way through the first few tasks, but you'll be much better served by asking questions early on and finding out how to approach your work process.

9.1 The First Month

Your first month will look really different depending on type of company you're going. Large and small companies, or more specifically, companies with large data science teams or small ones, will approach your onboarding from almost opposite perspectives. Let's take a peek at

what you expect from two companies, one that is massive with tons of data scientists, and one with no (or barely any) data science team. Back in chapter 2, this would be MTC and Seg-Metra respectively. These two examples highlight two ends of a spectrum, but the company you're working at will likely fall in between.



Onboarding at a large organization is like going through a factory line, while a small company is more ad hoc (Twitter Emoji from the Twemoji project)

9.1.1 Onboarding at a large organization: a well-oiled machine

You are one of the dozens of people starting this week. You get an email the week before telling you where to go, when to arrive, and what you need to bring. You then began a formal, multi-day onboarding process with people from all different departments. Together, you're issued your laptop and go through setting it up. You listen to presentations on the company culture, HR policies, and how the company is organized. Everything runs like clockwork - they have done this thousands of times before.

On the data science side, you'll get help setting up your coding environment. There's likely a checklist or extensive documentation on everything you need to do to get access to the data. There's a central repository of old reports and documentation of the data for you to read and absorb. No one is expecting you to deliver much right away – while they're excited to have you join the team, they know you'll need time to adjust. You'll be expected to take a few weeks to go through all the training and get your access approved for the systems. You may feel almost frustrated that it's taking so long before you feel productive, but a slow start that takes time to go through a process is natural in this environment.

If you are given a list of to-dos or an assignment, you should take it seriously but worry more about the process than the result. Established data science teams can often have their own idiosyncrasies that you'll need to adopt. It's not just good to ask questions at this stage,

it's essential to your ability to perform your job later on. The first few months are your chance to see what's been done before you and get yourself well versed in the rhythm of your peers.

9.1.2 Onboarding at a small company: what onboarding?

"Oh - you're starting today?" If you're joining a small start-up, don't be surprised if everything is not ready, including your laptop. You may be left on your own to figure out how to access the data. When you do get in, it may turn out it's not well optimized for your job and a SQL query on a small table of 100,000 rows takes 6 minutes to run. Onboarding sessions to learn about the company may not happen for weeks, if there even are any, because there aren't enough people starting in a given week for that to make sense to run them frequently.

There are no data science standards to speak of. No one is telling you what programming language to use or how to approach and structure an analysis. They are however asking you to quickly start getting results. Unlike at a large organization, you don't have to worry about not being productive—you'll be asked to do that right away. You do have to worry a lot more that you're accidentally doing something wrong and no one will tell you, or only finding out a few months later after your (incorrect) work is already being relied upon. That's why it's still imperative that you ask questions and work to get a foothold before you no longer have the fallback of being new. Going from crisis to crisis will cause you to burn out quickly, so work to build your own processes that will allow you to be successful in the long-run.

9.1.3 Understanding and setting expectations

One of the most important things you can do in your first weeks is to have a meeting with your manager to discuss priorities. This is important because it gives you the knowledge of what you are supposed to be working towards in your job, which allows you to work towards it. In some data science jobs, the priority is to provide analyses to a specific set of stakeholders to help grow a particular part of the business. For other data science jobs, the goal is to make high performing models that help run the business. For some jobs it's both or neither of these.

It may feel like you should already know what the job expectations are by the job posting and interview process. While this is sometimes true, a lot can change between the interview process and the job starting. The interviewers may not be on the same timeframe you're now working on, or the organization may have changed before your joined. By talking to your manager as early as possible, you'll get the most up-to-date information and have time to spend discussing it.

Ideally, your manager has a vision for what you'll be doing but is open to your priorities and strengths. Together, you want to define what success means in your job. Generally, that is tied to making your team and/or manager successful—if the members of the data science team aren't all working broadly towards the same objective it can be difficult to support each other. To define your own success, you need to understand what problems the team is trying to solve and how performance is evaluated. Will you be helping generate more revenue by

working on experiments to increase conversion, or will you be making a machine learning model to help customer service agents predict what concern a customer has, with the goal to decrease average time spent per request?

Performance usually does **not** mean "make a machine learning model with 99% accuracy" or "use the newest statistical model in your analysis." These are the tools to help you solve a problem, not the end goal itself. If your models and analyses are on problems people don't care about, they're pretty much useless. This is a common misconception from people entering their first data science job. It makes sense that it is common, since so much academic research and educational courses cover the many different methods of making accurate models. Ultimately, for most data science jobs having highly accurate models isn't enough to be successful. Things like the model's usefulness, level of insight, and maintainability are often more important. The next two chapters will go into these ideas in more detail.

You can't know, when you start a new job, what the expectations are in terms of job responsibilities. Some companies value teamwork and you may be expected to help with several different projects at once and drop your work on a moment's notice to help a colleague. Other companies ask that you have deliverables on a regular basis and it's ok to ignore emails or slack messages in order to achieve your project. The way you find out if you are meeting expectations is by having regular meetings with your direct supervisor. Most companies have a weekly 1 on 1 so that you can discuss what you are working on or any issues you have. These exist so that you can find out if you are spending your time on the tasks that matter to your boss. Why should you guess at what is wanted when you can get explicit feedback? Thinking in shorter term blocks will help you be sure that you are on the right track for when the larger performance reviews come along.

SETTING YOURSELF UP FOR SUCCESS

Unless you're at a very small company, there will be a formal performance review process, so be sure to ask about what that entails and when it will happen. One common practice is to have one every six months, with salary increases and promotions potentially following afterwards. Many companies do this as a "360" process, where you get feedback directly not just from your manager but also your peers. If this is the case, find out whether you will choose the peers or if they're chosen by your manager. This will help you understand who your most important stakeholders are.

For more established data science teams, there may be a matrix that says what areas you're evaluated on and what's expected for each of them at different levels of seniority. For example, one area could be "technical expertise." A junior data scientist may be expected to just have the foundations and show that they're learning, a mid-level that they have one area of expertise, and a senior that they're the go-to person at the company for a whole area, like A/B testing or writing big data jobs. If one doesn't exist, see if you can come up with a few areas with your manager.

Regardless of the system, make a plan with your manager to have a three-month review if it's not common practice. This review will help you make sure you're on the same page as your manager, give updates, and plan for the rest of your first six month and year.

The point of defining success is not that you already need to be excelling in every area in your first months; in fact, most companies won't do a formal performance evaluation of someone who's been there less than six months because much of that time has been on-ramping. Rather, it's to make sure as you learn about your role and begin work that you do so with the big picture in mind.

9.1.4 Knowing your data

Of course, you do need to learn about the data science part as well. If your company has been doing data science for a while, a great place to start is by reading reports they've written. Reports will tell you not only what types of data your company keeps and some key insights, but also the tone and style of how you communicate your results. Much of a data scientist's job is conveying information to non-technical peers and by reading reports, you'll have a sense of just how non-technical those peers are. See how simplified or complex certain concepts are and you'll be less likely to over or under explain when it comes time to write your own reports.

Then you'll need to learn where the data lives and get access to it. This includes knowing what table contains the data you want, but maybe also what data system has it. Perhaps the most frequently accessed data lives in SQL, but the event data from two years ago lives in HDFS (Hadoop Distributed File System) that you need to use another language to access.

Take a broad look at the data you are going to be working with on a regular basis but go in with an open mind. Some tables will have documentation (either packaged with the data or in a report about the data) which will explain potential quality issues or quirkiness. Read those first, as it will keep you from investigating "mysteries" that later turn out to already have been solved. Then take a look at a few rows and summary statistics. This can help you avoid "gotchas," where you find out that some subscriptions start in the future or that a column often has missing values. When you find these surprises that aren't documented, usually the best way to figure them out is to talk to the expert on that table. That may be a data scientist, if your company is large enough, or those who collected the data. You might find out that they're a true issue that needs fixing, or it might turn out to be expect. For example, subscriptions that start in the future could be those that have been paused and set to restart on that date. Or the coupons for last year's New Year promo you find used in May of this year is because support issued them.

Some companies are better than others about having data that was created for testing separate from real data, while others merge the data without a second thought. In the latter case you would want to ask around if there are orders or activity generated by test accounts or special business partnerships that you should exclude. Similarly, some datasets will include users with radically different behavior. American Airlines once offered a lifetime flying pass which included a companion fare. One of the people with the pass used the companion fare for

strangers, pets, his violin, and might fly multiple times a day. While you may not have anyone so extreme, it's not uncommon for newer businesses to offer deals that later look silly (e.g. 10 years of access for $100) which may need to be accounted for in your analysis.

Throughout this process of investigating the data you're figuring out what kind of overall shape your data is in. If you're at a smaller company, you may find that you need to work with engineers to collect more data before the overall data would be useful. If you're at a larger company, you'll be deciphering which of the dozens of tables to see if what you want exists. Maybe you're looking for tables with a column called "order" across 12 databases. Ideally, there should be well-documented, well-maintained tables for the core business metric, like transactions or subscriptions. But that is likely not to be the case for other less-important datasets, and you should try to learn more if you are going to be focused on one of the less documented areas.

Make sure you learn how the data got to you. If you're working with something like website data, it'll likely flow through multiple systems to get from the website to database you can use. Each of these systems likely changes the data in some way. When data collection suddenly stops, you want to know where to try and find the problem (rather than panicking). But some places will have data that people input manually, like doctors in a hospital or survey results. In these situations, you have to worry less about pipelines, and much more about understanding the many different attributes of the data and potential places where a human inputted it incorrectly. Pretty much anywhere you go you will have to deal with some data dirtiness.

As you go along, try writing down any "gotchas" in the data and a map of where everything lives. It's difficult to remember these sorts of facts over the course of a job, and many companies don't have great system for documentation or data discovery. Just like commenting code so your future self and others can understand its purpose, documenting data provides enormous dividends. While keeping this documentation locally on your laptop is okay, the best thing is storing it somewhere that everyone in the company can access. You'll be helping future new hires and even current data scientists at the company who aren't familiar with that specific area.

9.2 Becoming productive

Eventually you should be making your manager look good and lightening their workload, but at the beginning you will make it harder and that's expected. It will take you longer than you think to get fully productive. It's normal to feel frustrated during this time, but remember you're dealing with a lot of cognitive load from being in a new environment. You're trying to pick up on the (likely unspoken) norms about how long people take for lunch, what the working hours are, what forms of communication to use, whether everyone closes their laptop when they walk away from the desk, and more. On top of that, you have a whole data system to get to know.

It’s important to stress that it’s an easy mistake to make to assume that you have to prove yourself quickly and early, like "I need to do everything faster or else they'll wonder why they hired me". This is a case of impostor syndrome. Unless you're in a truly dysfunctional company, they'll expect some ramp-up time. Instead of proving yourself quickly, focus on positioning yourself to deliver value in the longer term (within months, not weeks). Early on, you're going to be asking more of the company ("can I get access to this, why is this query so slow") than you're getting back (in the form of reports and analyses).

That being said, you can still deliver some value early. Focus on simple and entirely descriptive questions, like "What is the distribution of our client sizes” or "what percentage of our users are active each week?" In the process you'll familiarize yourself with the company's data, but also find some of the snags and traps that are waiting. During your check-ins with your manager, show off some of your in-progress work so that you can see if you are headed in the right direction. It’s frustrating to put in a lot of time and find yourself answering the wrong question, using a methodology that your boss hates, or that you’re using the wrong data source.

Focusing on simpler questions also keeps you from embarrassing yourself by giving an incorrect conclusion because you are try answering a complicated question without learning all the details of the data first. This can be challenging because if your stakeholders are new to data science, their first question might be something like "predict which sales deals will close" or "how can we maximize user retention." But as we’ll talk about in Chapter 12, one of your jobs as a data scientist is to dig into the business question to find the data question behind it. If people don’t know or have misconceptions about basic facts (e.g. what percent of users make a second purchase, how many people click on ads), they won’t be asking the right questions.

There are two strategies that can help you become productive more quickly: asking questions and building relationships. Asking questions helps you more quickly understand the details of your job. Building relationships allows you to understand the context of your role in the organization.

9.2.1 Asking questions

One of the biggest things that can hold you back in your career is being afraid to ask questions or say I don’t know. As we’ve discussed before, data science is such a big field that no one knows everything, or even 20%! And there’s no way you could know all the intricacies of your company’s data. Your manager would much rather you ask questions and take up a few minutes of someone’s time then be stuck and spinning your wheels for days. Questions that are helpful can be anything from a technical question like: “what statistical test do we use for detecting a change in revenue in an A/B test?” to a business question like “which team is responsible for this product?”

That being said, all questions are not created equal. First, try to learn by observation about the question culture at the company. Do people ask questions in person, on a slack channel, in forums, or by email? Getting the channel right means you’re less likely to bother someone.

You also can ask your manager the meta-question of how to ask questions. Second, it's good to show you've been proactive. You might say "I researched this", "I'm excited to learn", or "this sounds like X, is it?" By having done some research yourself, you may be able to answer the question on your own and you will be able to ask questions with more of an understanding of the concept. Unless it comes up as you're working with someone or discussing an issue, you want to avoid asking questions that are answered by the first Stack Overflow result on google (e.g. "What's the difference between a vector and a list in R?").

While some of your questions will be more general, as the above, you might also have deeply technical questions. Finding out who the expert is on various statistical or programming methods is important as that person is generally the one you need to get answers from. You don't want to become a burden on this type of person (or anyone) so if you realize that you have many questions for a particular person, try to schedule a meeting with them. People are much less likely to feel put upon if they choose to schedule a meeting rather than facing questions every few minutes. It can also help to ask that person's style. Some people within the company have the role of supporting others, but if that person is also required to have deliverables, see if they have a calendar that blocks out certain times where they aren't reachable and respect that.

Third, avoid voicing criticisms veiled as questions, like "Why do you code this request this way instead of the clearly better way I learned in undergrad?" Try to genuinely understand why things are done the way they are. If the company's been around a while, there's a lot of technical debt – if they have physical servers, moving that to the cloud is a half-year plus of work for dozens of engineers. When people wonder "why don't we just do X, it's so easy and would save us a lot of time," they often assume it's because other people don't understand it's a problem or feel that it's urgent. But the reason they're not doing X may be because of things you have no idea, like legal constraints.

A great way to learn is by pairing with people. Instead of just asking questions and getting an answer, you can see how they found that answer. If it's a technical question, it's also a way to see someone's coding environment and learn new techniques. Even if your question is about how to get data, you can learn what table it is, how they knew which table, and maybe some coding tricks. Your eventual goal is to be able to answer as many questions yourself as possible by knowing where to look.

Finally, if you have questions that don't need an immediate response, try to keep a side list of things that might be useful to know (How often does the data refresh, what is the size limit for queries, how far back does x data go in the local server, etc.) and go over them in a block with your mentor or manager. This avoids the situation where you interrupt someone over and over, which can become a bother if you're not careful.

9.2.2 Building relationships

An important part of feeling comfortable in your new work environment is to build a support network. Some people do this more easily than others, but you want to make sure that you engage in some non-technical talk with people. In most cases this means setting up meetings

with people you've never talked to before to get to know them and their work. This isn't wasted time; it will allow both you and your coworkers to feel more comfortable relying on each other if you know more than your names and job titles.

Approaching someone you don't know can be daunting, but you can use your questions as a way to start a conversation with someone. People like being helpful and feeling knowledgeable so don't be afraid to use questions as an in as long as you are polite and friendly in your queries. Once you know a few people, even the largest offices feel less intimidating. It's also normal to message people you'll be working closely with to ask if you can set up a thirty-minute meeting to get to know each other. If you work in a large office, ask your manager to help make a list of people you should get to know.

At any sized office, it's good to find out who to go to for specific questions. Someone might be the company's best at SQL and someone else might be in charge of the experimentation system. It's very helpful to know who to turn to when you face a technical hurdle, and it usually isn't your manager. You also want to at least introduce yourself to your skip-level boss. This isn't so you can tattle on your boss, but rather because making yourself known to them will make it easier when they have to discuss you with your boss.

Similarly, you should try to meet with all the stakeholders you'll be working with. If the data science team is under 10 people, try to meet with them all individually. If there are data engineers or other data people you'll work with, talk with then. These can be informal, but it's important to not exist solely as an email signature. Even if you mostly work remotely, try to use facetime or skype so that people can see your face.

Do a lot of listening, both in official meetings and in social opportunities like over lunch. Meet people who work in data-adjacent areas (which could be everything from engineering to finance to sales ops to marketing analytics) and hear about how they currently do their jobs. Don't rush in with "I could do that better" or make commitments early like "we'll build you a machine learning platform to do that;" simply focus on collecting information and thoughts. And don't forget that everything doesn't always need to be about work. It's nice to get to know people on a personal level, whether by asking about their weekend plans, favorite tv shows, or their hobbies.

One last word to the wise: befriend the office manager. Office managers control a lot of the things that can make your day better: the snacks, the lunch order, what type of hand lotion is in the bathroom. They also have one of the hardest and most thankless jobs around, so make sure they feel appreciated.

Mentorship and Sponsorship

"Find a mentor" is one of the most common pieces of career advice, but it can be frustratingly in-actionable. It's true that having a mentor, someone who offer you career advice, can help you solve thorny issues and make better decisions. But unlike learning programming or improving your communication skills, there's no class you can take or book you can read that will get you a mentor. So how do you find one?

Fortunately, mentorship doesn't necessarily need to be a long-term relationship. Angela Bassa, who we interview in chapter 16, put together a list of people willing to answer questions and mentor data science newcomers at datahelpers.org. They may not be someone you can call with every career dilemma you face, but you could find one to help with a specific problem you're having, like practicing behavioral interviews or making your first R package.

But there's someone who can have even more influence over your career: a sponsor. A sponsor is someone who gives people opportunities, whether by funding their project, advocating for their promotion, introducing them to important people, or making sure they get assigned to the types of challenging projects that can help them grow. Especially for a sponsor, you need to show them you'll do a good job with the opportunity they're offering. For example, if someone recommends you to speak at a conference and you never respond to the organizer or give a clearly unprepared talk, that reflects poorly on the recommender. You don't have to have done the same thing before, but if you can show you did something similar (for example, gave a meetup talk) and you're responsive and polite in your communications with them, you can build their confidence you'll do the job well. Check out chapter 14 for advice on how you can start contributing to the community through open source, giving talks, or continuing to build your portfolio and blog.

If you want someone to be a long-term mentor or sponsor, keep them updated on how you've followed their advice or taken advantage of the opportunity they helped you with. Many people are mentors and sponsors because they want to help people, and it's gratifying for them to hear how you've benefited. And if you never communicate outside of when you need something, it can feel like you're just using them, which no one wants.

A lot of articles about sponsorship and mentorship are about finding one at your company, and this is especially important if you work at a large organization. But it's common for data scientists to change jobs every few years, and the data science community has enough small sub-pockets that you can start building a positive reputation and find sponsors and mentors who will stay with you through multiple companies.

9.3 If you're the first data scientist

Everything up to this point of the chapter was for the first few months of any data scientist position but being the first data scientist in an organization provides its own unique set of challenges. Given how new the field is and how many small companies do not have data scientists, being the first one isn't uncommon. Understanding that, as the first data scientist you should be especially prepared for when you begin.

When you start in your new position there will be absolutely no precedents. No one has decided yet if Python, R, or some other programming language should be used. No one has figured out how to manage the work—should software development practices like agile be used to decide what to work on, or should you do whatever you feel like that day? How should the code be managed: should a GitHub professional license be bought, a Microsoft TFS server be used, or can you keep all the files in your "My Documents" folder on your laptop without backups?

Because there are no precedents, everything you do will implicitly be a precedent. For example, if you happen to enjoy doing your work in the obscure programming language F#, then you are forcing the fact that the next data scientist will have to learn F#. It is in your best interest to make decisions that will be to the benefit of whatever the team looks like down the line. That may mean using a more common programming language than the one that is your favorite. This has to be balanced with the fact that focusing too much on the future can cause serious harm to the present. For example, if you were to spend 3 months

setting up a beautiful pipeline for automatically sharing reports with other data scientists but the second data scientist isn't hired for 5 years, then that work was a waste. Every day you will be either directly or indirectly making decisions with large consequences.

Besides having to figure out the role on your own, you also have to do the task of selling data science to the rest of the organization. Because the company hasn't had data scientists before, most people won't understand why you are around. The quicker people get your role, the more likely they are to want to work with you and keep a data scientist around. These conversations are also about managing expectations. As we've discussed before, some people think data science is basically magic, and their first data scientist can come in and immediately solve some of the company's biggest problems. You'll need to set realistic expectations about a) what data science is capable of and b) how quickly those goals can be reached. Therefore your job will require you to be constantly explaining data science in general to people, and also what in particular you can do to help the business. Sitting quietly in a corner working on models for months is possibly okay if you're the 20th data scientist on a team, but certainly won't work as the first.

While being the first data scientist is a lot more work and a lot riskier that other positions, it also has great payoffs. By making the technical decisions, you get to choose things that are more in line with what you want. By selling data science to the organization, you get well known and more influence. As the data science team grows you are in line to be the head of it, which can be great for career growth.

9.4 When it's not what was promised

Entering a data science job and finding it's nowhere near what you expected can be crushing. After months of work, you've finally broken into the field, and now you might have to go do it all over again. Worse still, you may worry if you leave quickly it will look terrible on your resume. Does that mean you have to put in a year? Managing a bad environment and deciding whether to leave is challenging. We'll cover two main categories of problems: that the work is terrible and that the work environment is toxic. While there's no silver bullet to solve these, we'll walk through some potential mitigation strategies.

9.4.1 The work is terrible

First, take a hard look at your expectations. Is the problem something along the lines of, "All my data isn't cleaned! I spent two days just on data prep! The data engineers don't fix everything immediately!" That's going to be part of every data science role. Even data scientists at the biggest companies with hundreds of engineers deal with this problem - there's so much data it's impossible for it all to be perfectly vetted. While the core tables should be clean and well-documented, you'll likely encounter data in your sub-areas you need to improve a lot or work with others to collect.

One way to check how realistic your expectations are is to check them with other data scientists. If you graduated from a related degree or a bootcamp, talk to your peers or people

in the alumni network about what they think of the data environment you're working in. If you don't know many data scientists yet, try to go to meetups or join online communities if you're remote (we'll cover this in-depth in Chapter 14). If there are other data scientists at your company who've had previous data science jobs, see if they can give you an idea of how this one compares.

Another situation may be that the work is tedious and boring. For example, the job you got hired to do might have been forecasting, but in practice all you do is hit the rerun button on someone else's existing forecasting model once a month. In that case, see if you can carve out some side projects in the organization or automate some processes. If it's boring work but not time-consuming, take the opportunity to do things that are data science related. Continue to build your data science portfolio of side projects and write blog posts or take online courses. These will help position you for your next role.

Finally, you can learn even from bad jobs. Is there any way you can tailor your job so that you do more things where you'll learn? What are the areas for you to improve upon? Maybe your peers won't help you learn how to write better code, but can you learn about pitfalls of building a data science team that are easy to make? It's very likely there are some smart and well-meaning people in your company, what happened to make it bad? By learning about what you *don't* like, you'll know what to watch out for in your next job search and be better prepared to avoid mistakes if you ever start your own data science team.

9.4.2 The work environment is toxic

The previous section covers a bad but manageable situation. But what if your work is really toxic? What if your manager and stakeholders have completely unrealistic expectations and threaten to fire you because you're not making progress on predicting lifetime value when they have no data for it? Or you get penalized when your answers don't meet the company's expectations? Companies newer to data science may expect you to sprinkle your data science fairy dust to solve the company's core problems. They may ask you, "Build a model to tell whether text is 'written well'," a problem no one in the field has even come close to solving. In this case, you need to adjust the expectations or risk feeling like a constant underperformer. Speaking up for yourself in this circumstance is difficult, but there are usually reasonable and smart people working for any company. If they say, "If you were a better data scientist you would be able to do this," that's a huge red flag. Even if a much more experienced data scientist could tackle the problem, that should have been recognized when they were designing the role and hiring.

Maybe the problem is instead that individuals or teams aren't collaborating. Instead of seeing where they can help, teams are constantly trying to sabotage each other. The focus is only on how they can get ahead, and they may even see it as a zero-sum game – if you or your team is doing well, that means that ours is losing. Besides creating an unhealthy environment, this often leads to a lot of wasted work as you may end up duplicating someone else's project because they wouldn't share their data or learnings with you.

The problem may have nothing to do with the data science part at all, but the environment is sexist, racist, homophobic, or otherwise hostile. There is no reason you should feel uncomfortable going to work every day. Even if it's not openly hostile harassment, getting constantly talked over in meetings, not having your correct pronouns used, or getting asked "but where are you really from?" all add up.

Unfortunately, these types of problems usually need the top leadership and active engagement from everyone to solve, but the fact that it exists is often indicative that leadership is non-existent or even actively contributing to the problem. If the problem is rooted in one bad person, hopefully others will recognize it and they will be removed, but if it's widespread it may be close to impossible to change and trying to do so as a junior employee is a quick recipe for burnout. In these situations, you want to think carefully about whether you need to leave.

9.4.3 Deciding to leave

Deciding whether it's right to leave your job is an extremely personal decision. While no one can give you a simple flowchart that will make the decision painless and easy, we can offer some questions to think about to guide your decision:

- Do you have enough savings, a partner with a second income that can support you, or family that you can ask for a loan from if you leave without having another job?
- Is your job affecting your health or life outside of work?
- If the issue is the work, have you talk with your manager about the issues and tried to resolve them?
- Is it possible to switch teams or roles, if not now than within a few months?

If the answers to these questions make you feel that you need to leave, one option is to immediately start looking for other jobs. But you may worry about how having a short job stint on your resume will look or how you would explain it to interviewers. If it's only been a few days or weeks and you came directly from your last job, consider getting in touch with your previous manager. It's likely they haven't filled your position yet and if you left on a good note, you may be able to go back.

If you are looking for new jobs, a few tips on how to handle talking about your short stint in your interview. First, wait for them to bring it up. Don't feel like you proactively need to talk about it – it may not be a concern, especially since they're clearly interested since you've got to the interview stage! Second, find some positive experience and learning from the job you can talk about – a project you worked on, the exposure to the industry, learnings from a senior leader. Finally, when asked about why you're leaving so soon, you can say something like, "The requirements of my job weren't what I expected, and I wasn't able to use my skills and expertise to benefit the company." If you learned something about the type of work environment you want, share it. For example, maybe you were the first data scientist at a company and you realized you want to be a part of a larger team. If you've decided to leave,

check out chapter 15 for more on how to do so gracefully, including searching while working full-time and leaving on a good note.

But maybe you can't leave, whether because your visa is tied to your workplace or it's the only company doing data science in your small town. If that's your situation, here are some tips:

- You are not your job. You don't have to take responsibility for poor decisions the company makes. Unless you're in a leadership position, you likely have little control over what the company does.
- Try to keep yourself healthy and not sacrifice sleep, exercise, and time with friends and family.
- Talk to someone about it. Maybe that's a partner, a friend, or a therapist. There may be some cases where they have advice, but just a listening ear will help.
- If you're experiencing harassment from a specific person, consider reporting it to your human resources department. Make sure you document your reporting - don't drop by HR in person; send emails and get emails back so that you have a record of the process. You may, eventually, want to point out certain things that were said to you, and having a written record will be helpful. If nothing is done, you can then file a claim with the Equal Employment Opportunity Commission if you're in the United States. Unfortunately, reporting harassment is not without risk: although it's illegal, companies have retaliated against employees for reporting by stalling their career growth or even eventually firing them. Even if you don't want to report it, consider keeping documentation of any harassment you experience in case you decide to at a later date.
- See if you can think outside of the box of how you can leave the company. Maybe you feel you "can't" leave because you don't want to have a short job stint on your resume, the only options available are to go to a less prestigious company or get a lower title, or you'd need to temporarily deplete your savings. But don't underestimate the negative effects of staying in a toxic environment – if you can make a short-term sacrifice to leave, it will likely be worth it in the long-term.

Hopefully you'll never be in this type of situation, but it's handy to have a little "break in case of emergency" information somewhere. Keep in mind that switching jobs in the data science field is common (as we'll discuss further in Chapter 15) so there's no career reason you should stay in a workplace which makes you feel uncomfortable.

9.5 Interview with Jarvis Miller, Data Scientist at Spotify

Jarvis Miller works as data scientist under the Personalization Mission at Spotify, focused on improving the listening experience for each user. When this interview was conducted, he was working as a data scientist at BuzzFeed. He graduated in 2018 with a master's degree in Statistics.

9.5.1 What were some things that surprised you in your first data science job?

Two things that surprised me were how much I could improve as a writer and how I needed to explain my data science contribution to the business without using jargon. I had this idea that since stakeholders had been working with data scientists, they have learned to understand the language and thus I didn't really have to change the way I explained things. I realized that that's absolutely not the case and I can't simply say, "I ran a logistic regression on this data to classify…" On the being a better writer side, I've begun to flesh out the story when I write a report, improve my data storytelling abilities, and explain things in a way to where product managers, designers, and stakeholders who aren't in tech at all can understand what I'm saying.

I came from academia, where I felt it was all about whether you found the result at the end of the day - it didn't matter whether you started working right before the deadline or if you planned way ahead. In industry, you have a big overall goal but you figure out how to break it down into versions. You get the first version working, ship it, learn about whether it's doing well or not, and maybe improve it in a future quarter. I was used to going until it is done. But here, I had to learn how to prioritize parts of a project and then wrap it up. I'd document what I have done, what there is to do in the next version, and make it shareable, whether that's by putting a report into a shared folder or making an app so people can use it and see what is supposed to do.

9.5.2 What are some issues you faced?

Speaking out was something I really struggled with. When I started, I was doing an isolated project and the person I reported to was in New York and I was in LA. If I was confused, I didn't know whether I should message them immediately or save it for our meeting. I knew I didn't want to be derailed by something that was blocking my work, but I wasn't even sure when something is a blocker. I think this is a common problem for data scientists, especially from those in marginalized groups or who switched from a different field. They may feel like that because they're new or not experts, they can't express displeasure or voice an opinion. If I could go back, I would have a conversation sooner about how I'm feeling isolated and how I'm not sure how communication works.

9.5.3 Can you tell us about one of your first projects?

One of them was to revamp our A/B testing platform, which was a very broad problem. I started by getting a list of people to talk with about what they did at Buzzfeed, how they worked, and how A/B testing fit into that workflow. We then discussed the specific tool – what did they dislike and why, and what was their workflow when using it? Unfortunately, this led to the issue of taking on too much. A lot of people had multiple suggestions and I gave them all equal weight, which ended up with fifty big things that I needed to do. But my manager asked me to break those suggestions up into the must haves and the nice to haves, including the reasons why these those were prioritized are the way they are. He suggested listing the

overall goal of the project and giving ideas weights based on their contribution towards the goal and how long they will take.

9.5.4 What would be your biggest piece of advice for the first few months?

Remember that you've been hired for a reason - they respect your point of view and they think they can help you learn and that they can learn from you. If you have an opinion, try and let someone know. If you hate speaking in a big group, maybe message one person, run it through with them, and bounce back and forth on ideas before you express it in front of the larger group.

This doesn't just apply for the technical side of the job. For me, in the first few minutes of my one-on-one with my manager, I don't want to immediately jump into what I've done. I want to have a few minutes to have a casual conversion to destress and clear my mind. I know that this helps my productivity and my company wants me to be productive, so I should let them know. Your opinion is valued and it's worth sharing, especially if it's about how you like to be treated or how you can be most productive and flourish in this role, because they don't know you like you know yourself and your being productive will benefit everyone.

9.6 Summary

- Don't worry about becoming fully productive right away – instead, focus on building relationships, tools, and your understanding of the data, which will make you productive in the long-term
- If you're in a bad work situation, try to work where you have control to mitigate the impact on your health and career

10

Making an effective analysis

This chapter covers

- Planning an analysis
- Thinking through code, data, and project structure
- Delivering the analysis to the client

The chapter is written in the context of data scientists who focus on decision science and analytics—people who use data to provide ideas and suggestions to the business. While machine-learning engineers have to do analysis too before building and deploying models, some of the content around stakeholder management with pretty visualizations is less relevant. If you're a machine-learning engineer and you're reading this, don't worry, this chapter is still plenty relevant to you and you'll love the next chapter on deploying models into production.

The backbone of many data scientist's jobs is making analyses: brief documents that use data to try and explain a business situation or solve a business problem. Modern companies are built on reporting and analyses—people who make decisions aren't comfortable doing so without data to back up their choices, and data scientists are some of the best people to find meaning in the data. Analyses are also important for building machine learning tools, since before a machine learning model can be built, the context of a dataset has to be understood through an analysis. Crafting an analysis that can take the vast amount of company data and convert it into a concise result that clarifies the matter at hand is extremely difficult and practically an art. How can a person be expected to take tables with millions of records on historical information, each with complexities and nuances within them, and turn that into a definitive: "Yes, the data says this idea is good?" The act of figuring out what is meaningful mathematically, what the business cares about, and how to bridge the gap between those is not something you should expect to know how to do naturally. In this chapter we'll go through

the basics of how to build an analysis so that you understand how to provide meaningful analyses to the company. By using the skills in the chapter, you should be able to more quickly grow in your data science career.

What is an analysis, really? An analysis is typically a PowerPoint deck, a PDF or Word file, or an Excel spreadsheet that can be shared with non-data scientists containing insights from data and visualizations displaying them. See figure 10.1 for an example of slide that could be found in an analysis. An analysis generally takes on the order of 1-4 weeks to make, with a data scientist having to collect data, run code for statistical methods on it, and make the final result. Once completed the code is not touched until the analysis must be rerun months later, or possibly never. Some example analyses include

- Analyzing customer survey data to see which products have highest satisfaction
- Looking at data on the locations that orders are being made from to pick out the location of a new factory
- Using historic airline industry data to predict which cities will need more routes to them

These examples have varying levels of technical complexity—some require only summarizing and visualizing data while others need optimization methods or machine learning models, but all of them answer a one-time question.

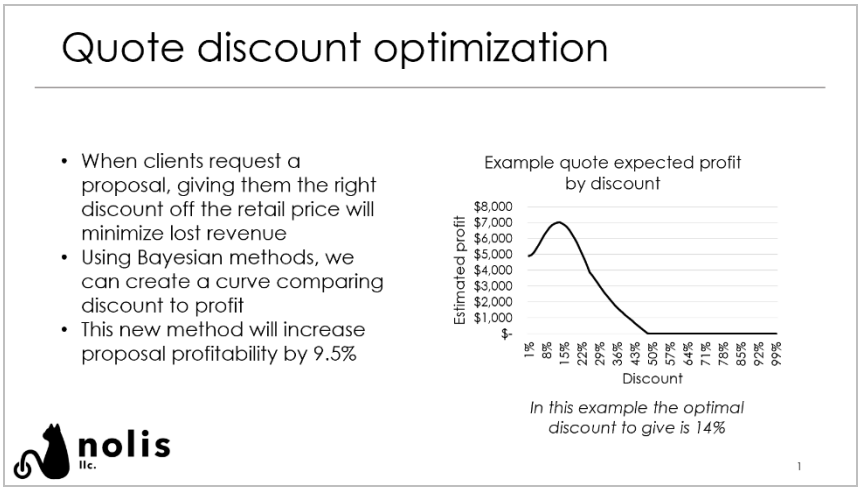


Figure 10.1 An example slide of an analysis PowerPoint deck.

Reporting versus making an analysis

A report and an analysis are similar, but not the same thing. A report is something that is generated on a recurring basis without much structural change between versions. For example, the monthly financial report may be a large Excel spreadsheet that updates with new numbers each month. The point of a report is to keep people aware of how metrics are changing. An analysis is something that is done on a one-time basis to answer a deeper question. For example, a

customer acquisition analysis may be work done in R on how new customers are purchasing products, with the results put into a PowerPoint. Reports tend to be filled with numbers and metrics, where analyses focus on providing a single main result. Most of the traits of a good analysis hold for a good report, so in this chapter we use "analysis" to mean both unless explicitly stated otherwise.

So, what makes a good analysis? A good analysis has these five traits.

- **It answers the question** – an analysis starts with someone asking a question, so for the analysis to be meaningful it needs to provide an answer. If the question proposed was: "which of these two websites cause more customers to buy products?," then the analysis should provide an answer to that. That answer could even be "we don't have enough information to know," but it must be a direct response to the question.
- **It is made quickly** – Answers to business questions will influence decisions that have deadlines. If the analysis takes too long to create, then the decision will be made without the analysis. A common expectation is that the analysis will be complete within a month.
- **It can be shared** – The analysis needs to be shared: not only with the person who asked for the analysis to be done, but also with whoever that person wants to share it with themselves. That means if the analysis involves a plot, the plot can't just live within an R or Python script, it has to be in a format people can digest—like a PowerPoint or HTML file.
- **It is self-contained** – Since you can't predict who will see the analysis, it needs to be able to be understood on its own. That means plots and tables have to have clear descriptions, axes should be labeled, explanations in the analysis should be written down, and it should avoid referencing other work if possible.
- **It can be revisited** – Most questions will be asked again in the future. Sometimes that means doing the exact same work again, like rerunning a clustering. Sometimes that is using the approach somewhere else, like changing the input data from European customers to Asian customers.

These traits aggregate up to a general theme of "a good analysis is something that helps non-data scientists do their job."

The rest of this chapter is structured to cover the steps of an analysis chronologically, starting with the initial request for the analysis and ending with the report being handed over. While not every analysis will follow these steps, most will (or should). As you become more familiar with making analyses you may feel inclined to skip some steps, but those shortcuts are the very actions that cause senior data scientists to make mistakes. Treat this as though it were a checklist for an aircraft pilot.

Analyses for different types of data scientists

Depending on your role as a data scientist, the times when you'll be doing analyses will vary greatly:

Decision scientist – for these types of data scientist, making analyses is the core function of the job. Decision scientists are constantly diving into the data to answer questions, and those questions need to be communicated to the business. An analysis is the key tool for doing so.

Machine learning engineer – while a machine learning engineer has a focus on creating and deploying models, analyses are still useful tools for sharing how well the models are performing. Analyses are used for show the value in building a new model, or for seeing how the models change over time.

Analyst – analysts, data scientists who focus heavily on metrics and KPIs for the business, will usually find themselves making lots of reports. They will create a stream of recurring data for the company, often in Excel, SQL, R or Python. While these analytics experts will make analyses, they more than the other roles need to think about maintainability of the work since they have to repeat it so much.

10.1 The request

An analysis starts with a request for an answer to a business question—either a person from a different part of the business or your manager will come to you with a question like, "Can you look into why the widget sales were low in Europe in December?" or, "Do our small business customers behave differently than our larger ones?" Depending on the level of technical expertise of the person asking, you may get an ill-defined request ("why are sales down?") or a precise one ("what attributes are correlated with a lower average order value?").

The analysis is formed around the business question, but you can't do data science on a business question. Data science questions are things like "how do you cluster these datapoints" or "how do we forecast sales?" The data scientist has to do the job of converting that business question into a data science question, answering the data science question, and returning a business answer. This is tricky to do—understanding how data science questions and business questions relate is a combination of experience with the type of problem at hand and an understanding of how the results of different statistical methods could potentially be useful.

Figure 10.2 shows this process graphically. The business question comes from the stakeholders; for example, they may want to know how to target marketing to different customers. The data scientist has to figure out what that means in mathematical terms; in this example, that might be a clustering of the customer data. When the process is complete the data scientist has a data science answer (like a set of three groups of clustered data points). Finally, the data scientist has to convert that back into something the business would understand, like groups such as "new customers" or "high spenders."

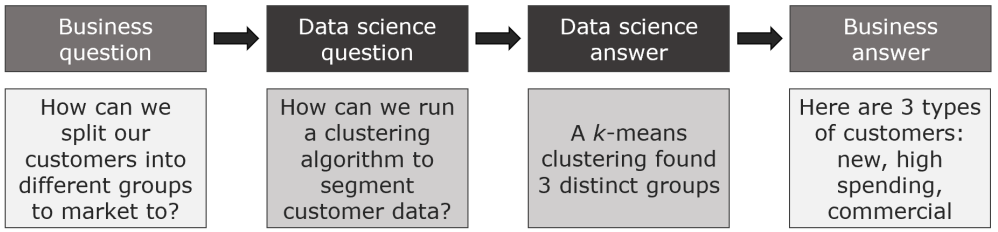


Figure 10.2 The process of answering a business question with data science, devised by Renee Teate

So, before you start looking at data and writing code to solve a data science question, there is foundational work to be done to best understand the business question. You need to understand what the context is for the analysis, so you can best deliver something helpful. Who is asking for the analysis to be done, and what is their relation to their team? What is their motive? Do they have a very specific question they want answered, or a vague general idea of a problem and a hope that data could be useful for it? Does it seem like you might even have the data to solve this problem, and if not, what would it take to get it? Not only does asking questions help you understand how to solve the problem, it also helps you understand what it will be used for—many data scientists have spent weeks on analyses only to discover there was no need for it and the stakeholder was "just curious."

These questions are generally answered with a 30- to 60-minute kickoff meeting with the person making the request, plus anyone else involved in the work. As the person doing the analysis, you may not be the one to organize a meeting like this, but if you don't seem to have one on your calendar, it is worth it to schedule one. If you haven't met the person requesting the analysis before, this meeting is a good time to make an introduction and learn about their work.

A hypothetical example set of foundational knowledge would be something like:

- Who is requesting the analysis: Julia from the widget product team
- What's the motive: widget sales are down 10% this month and they don't know why
- What's the request: use data to see if the widget sales dip was focused in one part of the country
- What decision will be made: whether the widget product should be discontinued
- Do we have data? Yes, customer orders by shipping zip code is available

Knowing if you have data that could plausibly answer the question is *really* important. The last thing you want to do is spend several weeks working on an analysis, only to have to return to the stakeholder without anything they could use. An example of a situation where you wouldn't have data would be something like the following: at a retail company, the stakeholder wants to know how many orders each customer made, but since customers pay in cash, there is no way to use existing data to tell who made each order. In these situations, it's best to be up front with all people involved and let them know what they are asking for isn't

possible. Other people may propose alternative ways to use data that may be close enough to what you hoped for, or you may have to explain why the alternatives won't work either. If at all possible, try to propose a plan that could, someday, get the required data. In the case of the previous example, a loyalty program would allow orders to be associated to a particular customer and fix the data problem, although that would take time to make.

The other questions, such as who the person is and why they are making their request, will be useful when creating the analysis plan.

10.2 The analysis plan

As a data scientist, nothing is more fun than diving right into some data to answer questions. Let's load data! Group it! Summarize it! Fit a model and plot results! Unfortunately, with there being an infinite number of ways to summarize and model data, you can spend weeks working with the data only to discover that nothing you've produced actually answers the business question that was proposed. The realization that you haven't done something relevant is *the worst*. And it happens frequently to data scientists, especially junior ones who haven't been burnt too many times yet.

One solution to this problem is to have a guardrail in place that keeps you on track and doing work that's relevant. An analysis plan is that guardrail. The idea is that before you start looking at data, you first write down everything you plan to do with the data. Then as your analysis progresses, you keep track of how much of the plan you've completed. When you've done everything on the plan, you're done! So not only do you have a way of knowing if you're off the plan, you also have a tool for tracking progress and keeping yourself accountable. You can even use it in meetings with your manager to discuss how things are going.

When making an analysis plan, you want to have the work in the plan be actionable—"make a linear regression on sales by region" is something you can write code to do, whereas "find out why sales are down" isn't something you can just do, it's the result of doing other things. By having the tasks in the plan be actionable, it'll be easy to tell if you are making progress. It will also make doing the analysis easier because you won't have to worry about what to do next. Instead, you'll be able to just look at the analysis plan and select the next task on there to do.

For making your first few analysis plans, we strongly recommend the following template:

- **The top:** put the title of the analysis, who you are (in case it's shared with others), and the objective of the analysis.
- **Sections:** each section should be a general topic in the analysis. The analysis work done within each section should be self-contained and not rely on the work of other sections, so that it should be possible for each section to be done by a different person. Within each section should be a list of tasks to do.
- **First level of section lists:** the first level of the section lists should be each question that was posed. This will help everyone remember why you're doing that specific work, and if all the questions are successfully answered the topic of the main section should

be considered understood.

- **Second level of section lists:** the second level of the lists should have the actual tasks to do that can be checked off as the work is being done. These can be tasks like types of models to run, and the descriptions should be specific enough so that at any time you could concretely say if the work has been completed or not.

Figure 10.3 shows an example analysis plan, in this case for assessing why customers are leaving in the North American region. At the top is the title, objective, and contact information of the data scientist in case the material is passed around. Each section of the plan covers a different component of the analysis (e.g. either analyzing within North America or comparing to other regions). The subsections (numbered) are questions in the analysis, and the lowest section (lettered) are the specific tasks to be done.

Figure 10.3 An example analysis plan

When you make your analysis plan, share it with your manager and the stakeholder making the request. They should either give suggestions on how to improve it or approve of the work. Having approval on the analysis plan provides an agreed-upon foundation for the

work; if, after you do the analysis, the stakeholder asks why you did it that way, you can refer to the analysis plan and the original goals.

It's likely that as you do the analysis, you'll realize you left something important off the analysis plan or have a new idea that you hadn't considered before. That's totally fine—just update the plan and communicate with the stakeholder that you're making the change. Because you have time constraints, you may have to remove a less important task from the existing plan, but again the analysis plan is useful because it will create a conversation around what to remove instead of making you attempt to do an impossible amount of work.

10.3 Doing the analysis

With signoff on the analysis plan, you can actually get started on the analysis itself! The work starts with importing data so you can manipulate it and cleaning it up. Then you repeatedly transform the data by summarizing, aggregating, modifying, visualizing, and modeling it. When ready, you communicate that work to others.

Here we'll briefly cover some of the considerations you should have while doing an analysis in a work environment. There are whole books dedicated to this topic that also teach you the code for conducting the analysis in the language of your choice.

10.3.1 Importing and cleaning data

Before you can tackle the questions on your analysis plan, you need to have the data in a place where you can manipulate it and in a format you can use. That usually means being able to load it in R or Python but could include using SQL or other languages. Almost always this will take you more time than you expect. There are many surprises that can appear in the process. A few of these many horrors include:

- Issues with connecting to company databases in your particular IDE
- Issues with incorrect datatypes (such as numbers as strings)
- Issues with weird time formats ("year-day-month" instead of "year-month-day")
- Data that requires formatting (like if every order id started with "ID-" and you need to remove that)
- Missing data

What's worse, none of this work looks productive to non-technical people; you can't show the stakeholder a compelling graph of how you got a database driver working, nor would they understand that string manipulation helps them with their business problem. So as tedious as this task is, you want to get moving to doing the data exploration quickly.

When working through importing and tidying the data, consider that you have a dual mandate: spend as little time as possible on anything that wouldn't be needed, and as much time as possible on work that helps down the line. For instance, if you have a column of dates that are stored as strings and you doubt you'll ever need that column, don't spend time changing the strings to the right datetime format. On the other hand, if you do think you'd

need that column, definitely do the work as soon as possible since you want to have a clean set of data for your analysis. It's hard to tell in advance what will be useful, but if you find yourself spending a lot of time on something, ask yourself if you really need it.

When importing and tidying data, you may find yourself stuck for multiple days on a single issue like connecting to a database. If you find yourself in this situation you have three options: (1) ask for help (2) find a work around, and (3) keep trying to fix the issue yourself. Option (1) is great if you can do it: a senior person may be able to find a quick fix and you can learn from what they did. Option (2) is great too, doing something like using a flat csv file instead of a database connection will get you going to the analysis which provides the business value. Option (3), keep trying and trying, is something you should avoid at all costs. If you spend days and days on a single issue, it will make it look like you aren't able to deliver value. If something is insurmountable to you, then discuss with your manager what to do—don't just keep trying and hope that somehow it'll all resolve itself.

Once you've loaded the data and formatted it, you may start using it and find *weird data*. Weird data is anything that is outside basic assumptions. For instance, if you were looking at historic airline flight data and found some flights that landed before they took off—well that would be weird since generally airplanes take off first! Other weirdness could be anything from a store selling items that had a negative price to manufacturing data where one factory made a thousand times more items than an otherwise similar factory. These sorts of weird artifacts show up all the time in real-world data and there is no way to predict them until you look at the data yourself.

If you find yourself in a situation where you have weird data, don't ignore it! The worst thing you could do is assume the data is fine, and then after weeks of analysis work find out that the data wasn't fine and have your work go to waste. Instead, first talk to either your stakeholder or someone responsible for the data you're using and see if they are aware of the weirdness. In many cases they may already know about it and suggest you ignore it—for example in the previous situation where you are look at airline data you may be able to just remove the data for flights that landed before taking off.

If it turns out that the weirdness was unknown and could jeopardize the analysis, you then need to investigate if there is any way to salvage it. For instance, if you were going to do an analysis comparing revenues to costs, and weirdly half of your data is missing costs, you'll need to see if you can work with just the existing costs or just with revenue alone. In a way this becomes an analysis within an analysis—you're making a mini-analysis to see if the original analysis is even feasible.

10.3.2 Data exploration and modeling

During the data exploration and modeling part of the analysis, you go through the analysis plan point by point and try and complete the work. Here is a general framework for how to tackle each point.

USE GENERAL SUMMARIZATION AND TRANSFORMATION

The vast majority of analysis work can be completed by summarizing and transforming data. Questions like "how many customers did we have each month" can be answered by taking customer data, grouping it at the month level, then counting the distinct number of customers in each month. This requires no statistical methods or machine learning models, merely transformations.

It is easy to view this as not really "data science" since it doesn't require anything beyond a lot of arithmetic, but often doing the transformations in the right way is immensely valuable. Most other people in the company either don't have access to the data in the first place, don't have the ability to do the transformations effectively, or wouldn't know the right transformations to do.

Depending on the data, you may want to throw some statistical methods in, like finding values at different percentile levels or computing a standard deviation.

VISUALIZE THE DATA OR CREATE SUMMARY TABLES

After doing the appropriate transformations, create visualizations or summary tables of it to better see what is happening in the data. Continuing the example from before, if we had the number of customers each month, we could create a bar plot to see how they have changed. This can make it easy to see what patterns are in the data in a way that you couldn't by merely printing a data frame to a screen. Figure 10.4 shows an example of a summary visualization showing the overall customer count each month. With this graph, people can easily see that the customer count is trending slightly upwards.

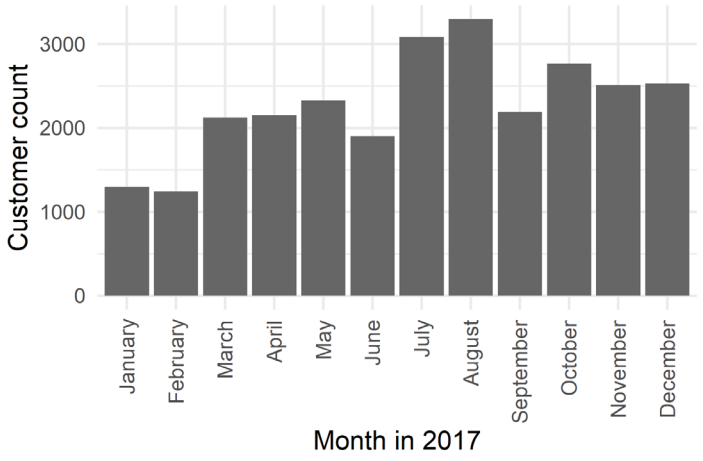


Figure 10.4 An example summary table

The actual visualization you choose depends highly on the data at hand. You may want to use a line graph, a box plot, or any one of many possible choices. You may also make summary data tables instead of a chart, depending on what you are trying to understand. Consult the appendix for resources on how to choose the right kind of plot for your data. Note that as you make the visualization, you may realize you need to change some steps in the transformation before. You'll likely go back and forth many times between these steps.

Since you'll be iterating through visualizations and continuously transforming the data, you'll have to balance your desire to delete the mediocre ones to keep your code clean with a desire to save everything just in case you need it again. The best practice is to save as much as possible provided (1) your old code doesn't break after you make further changes (2) you can clearly denote which results are "the good ones." Avoid keeping code that doesn't work in your analysis, or massive areas of code that are commented out—those situations make maintaining the code enormously difficult. All of this is further enchanced with using version control like git and GitHub—by continuously making commits every time you add new content to the analysis you'll be able to keep a log of what you've done and roll back code that suddenly breaks things.

CREATE A MODEL AS NEEDED

If you see patterns in your data that suggest modeling is a good idea, do it! For example, maybe it makes sense to apply a time-series model to the customer counts to predict next year's customers. When creating models, you'll want to output results and visualize them to understand how accurate or useful the models are. For instance, create plots that compare the predicted results to the actuals, or show metrics like AOC scores and feature importance.

If you create machine learning models that may be used outside of the analysis, like potentially being put into production (which the next chapter will cover), make sure you are isolating the code that builds the model from the general analysis work. Since in the future you'll want to use only the model, you'll want to easily be able to pull that code out from the code that makes the general visualization charts.

REPEAT

You should complete these steps for each point of the analysis plan. During the steps you may have a new idea for what to analyze or realize that what you thought was a reasonable question doesn't make sense. This is when you should adjust your analysis plan and continue on with your work.

It's likely that the different analysis plan points related to each other, so that the code you used to done one point will be repeated in another. It's worth it to put effort into structuring your analysis plan so you can run the same code repeatedly, and updates to one part of the plan instantly roll out to the others. Your goal is to make a set of code you can maintain—you can easily modify it without spending a ton of time keeping track of complex code.

10.3.3 Important points for exploring and modeling

The work of data exploration and modeling is extremely dependent on the problem you are trying to solve. The mathematical and statistical techniques you'd use for trying to cluster data are quite different from making a prediction or trying to optimize a decision. That being said, there are some broad guidelines that can help to make the difference between an okay analysis and a great one.

FOCUS ON ANSWERING THE QUESTION

As brought up with the analysis plan, it's extremely easy to waste time by doing work that doesn't support the goal. For example, if you're analyzing customer orders to see if you can predict when a customer will never come back, you may get a neural network model working decently, and then spend multiple weeks tuning the hyperparameters. However, if the stakeholder only wants a yes or no on if it's feasible in the first place, then tuning hyperparameters to make the model slightly more efficient doesn't help them. The weeks spent on hyperparameter tuning could have instead been spent on something more relevant.

When doing the analysis, it's important to continuously stay focused on the analysis plan and answering the question the business asked. That means constantly asking yourself "is this relevant?" That should be something you consider every time you make a plot or table. If you find yourself repeatedly thinking what you're doing is relevant, that's great. In the much more likely instance where you find yourself occasionally thinking, "This plot (or table) isn't useful," then you may have to adjust your work. First, try stopping what you're doing and taking a different approach to the problem. For example, if you were trying to group customers by their spend deciles, try instead doing a clustering. By taking a dramatically different approach you're more likely to succeed than changing what you're doing only slightly. Second, talk to your manager or the project stakeholder; it could be that the data you are using isn't effective to solve the problem at hand.

Over the course of the weeks you do the analysis, you should be steadily building up a collection of results that are truly relevant, and hopefully following the analysis plan.

USE SIMPLE METHODS OVER COMPLEX ONES

Complex methods are so exciting! Why use a linear regression when you can use a random forest? Why use a random forest when you can use a neural network? These methods are shown to perform better than a plain 'ol regression or *k*-means clustering, and they're more interesting. So, when people come asking to solve business questions with data, surely we should deliver the best methods possible.

Unfortunately, complex methods come with many drawbacks that aren't visible from focusing solely on their accuracy. When doing an analysis, the goal isn't to get the best possible accuracy or prediction, it's to answer a question in a way a businessperson is able to understand. That means you need to explain why you got the result you did. It's easy with a simple linear regression to provide plots of how much each feature contributed to the result,

whereas with other methods it can be very difficult to describe how the model produced the result. This will make it harder for a businessperson to believe your results. More complicated methods are also more time intensive to set up—it takes a while to tune and run a neural network whereas a linear regression is pretty quick.

So, when you're doing your analysis, as much as possible choose simple methods, both in models and in transformations and aggregations. For instance, rather than pruning some percent of outliers, doing a logarithmic transformation or taking the median instead of the mean. If a linear regression works reasonably well, don't spend time building a neural network to slightly improve accuracy. This will make the result much easier for other people to understand and much easier for you to defend and debug.

CONSIDER PLOTS FOR EXPLORATION VS PLOTS FOR SHARING

There are two different reasons that a data scientist would choose to visualize data: for exploration and for sharing. When making a plot for exploration, the point is to help the data scientist understand what is happening in the data. Having a complicated and poorly labelled graph is fine so long as the data scientist understands it. When making a plot for sharing, the goal is for someone who doesn't know much about the data to get a specific point a data scientist is trying to make. Here the plot must be simple and clear for it to be effective. When making an analysis, you should use lots of exploratory plots, but those shouldn't be used for sharing.

Let's take an example based on fictional data about pet names in a city, where a data scientist wants to understand if the letter a pet name starts with relates to the species of the pet (cat vs dog). The data scientist loads up the data and makes this visualization, showing for each letter, the split of cats vs dogs that have that as the first letter in their name. This can be seen in Figure 10.5:

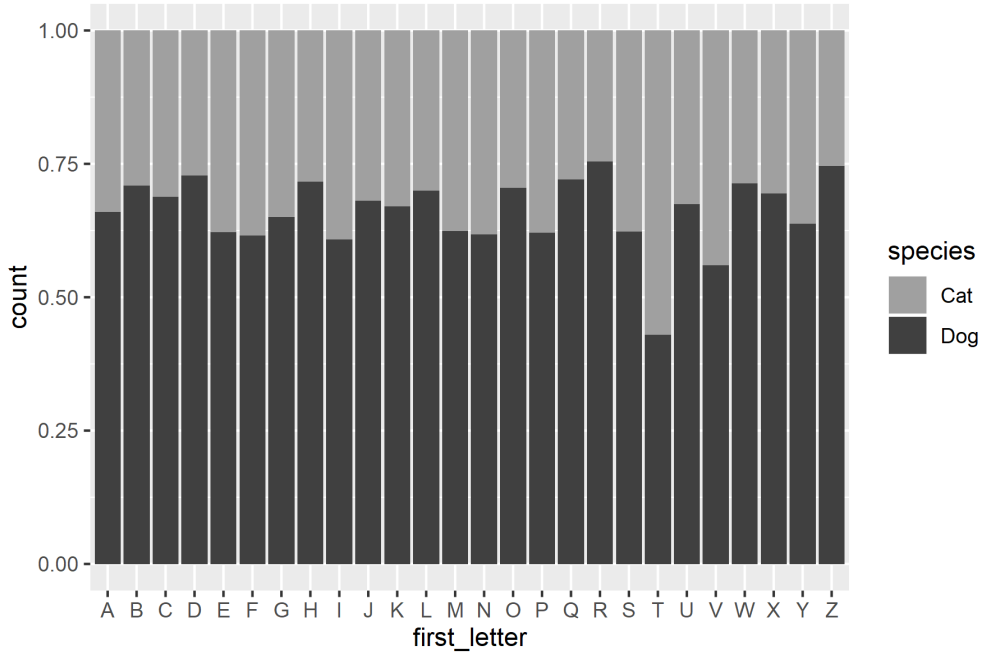


Figure 10.5 Example of a visualization made during an analysis before cleaning

If you look at Figure 10.5 you'll notice that the "T" bar has a much higher amount of cats vs dogs—a meaningful finding for the data scientist. That being said, this plot isn't something you'd want to show to a stakeholder—there is lots going on in the plot and it isn't clear what the point is at first glance. Figure 10.6 shows the same data plotted in a different, more sharable way. In this version, it's clear that cats have a 15% chance of a name staring with T while dogs only have a 5% chance. This same data is now something that could be shared.

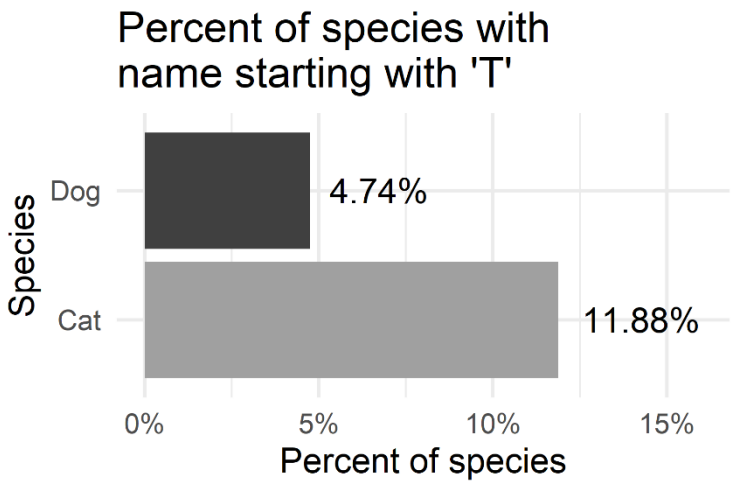


Figure 10.6 The same data as Figure 10.5, but plotted to highlight the importance of the letter "T"

CONTINUOUSLY READY TO SHARE

The result of the analysis can take different forms, and which form to choose is typically based on the target audience. If the analysis is to be handed to businesspeople, then a slide deck or editable document is often used. PowerPoint or Word (or Google slides or Google docs) is a good choice since anyone can view it (as long as they have the Microsoft office suite for the first two), and it can include lots of charts, tables, and text descriptions. If the analysis is for technical people, you could hand over a Jupyter notebook or Rmarkdown output HTML file. These methods are good because they usually require less work to polish (e.g. you don't need to spend time aligning figures on a slide). If the analysis requires handing over many tables of data to financial people, Excel may be the best choice. Excel is a great tool when the end user will need to take the numbers in the results and do further calculations on them. You should decide early in the analysis-making process what kind of output you expect to deliver to avoid rework later.

Depending on how big the scope of the analysis is, you'll want to periodically check in with the person who you are making the analysis for and show them your work. This is to avoid the terrible situation where you spend weeks working on an analysis in isolation, and when it's time to hand it over the stakeholder points out something that invalidates all of your work (e.g., "You looked at customer sales but forgot to consider returns."). In these situations, had that been pointed out in the beginning, you'd be able to avoid having to throw away lots of work. In addition to avoiding bad situations, the stakeholder can often contribute by suggesting possible areas to focus on or methods to try. In a sense, checking in with your stakeholder throughout the making of the analysis is similar to the software development

concept of agile: continuously putting out improvements to the work rather than making one massive software release.

Frequent check-ins with stakeholders are great, but data scientists often neglect to do them. The downside of having a check in with someone is that the work has to be able to be shown to a non-data scientist—it has to be at a sufficient level of polish that it's not embarrassing to show. Things like plots with clear labels and meaning, code with minimal errors, and a basic story behind what's happening are all required. So, it's easy as a data scientist to think, "I'll put off sharing my work until I polish it, and I'll put off polishing it until later." Don't do this! It'll almost always end up being more work in the long run. By continuously maintaining a level of polish so that you can share your code, you will end with a better product.

ONE BUTTON RUN

Just like how it should only take running one script to load and prepare your data, your analysis should take one button press to run too. This means in Python having a Jupyter Notebook that will automatically load the data and do the analysis without error. In R, have an Rmarkdown file that will load the data, analyze it, and output a HTML file, Word document, or PowerPoint deck.

When doing the analysis, you'll want to avoid too much running of code outside of the script or running your scripts out of order. These practices make it more likely than when you rerun the whole script it will have an error. It's okay to do ad-hoc coding a bit, but just continuously make sure you can rerun your file without errors. This will help keep your results continuously ready to share with other people, and make you spend less time fixing the script at the end of the analysis.

10.4 Wrapping it up

Depending on the particular stakeholder for this analysis, the output of your code may be enough to satisfy the request, or you may have to go further and make a final version. If a polished, final version is required, such as a PowerPoint presentation, you may need to do a final level of polishing beyond what you did while making the analysis so that you adhere to company style guidelines. And importantly you'll need to craft a narrative for the final document, so that people not involved in the work can fully understand the conclusions of the work, what was done, and why.

Making that narrative is the first step for a good final document—what kind of story are you going to tell in this? How are you going to introduce the problem, explain how your work provides a solution (or doesn't), and the next steps? There are many ways to create a narrative, but one simple one is to think about how you would explain this out loud to a person who hasn't seen your work before. Think about the story you would tell them and try and tell that through your document. Repeatedly ask yourself, is what I am showing going to

be understandable to my audience? What can I do to improve this? Eventually you'll get to a point where you are happy with your content.

You'll also need to add text to your document—usually to explain the narrative you have or why each graph is worth sharing. Again, try to make it understandable to someone who doesn't have the context you have. Have the text answer the question: how is what I am showing useful for the business? Different companies have different standards around how much text to include—some companies want detailed descriptions explaining everything, while other companies are happy with a few words. Try and air on the side of overexplaining since you can cut content later.

Once you think your material is ready, you'll want to have some peer review to check for small mistakes before sending it to the stakeholder. Consider having someone on your team who is familiar with the context of the work check it to see if everything makes sense. Depending on your company your manager may require you do this with them so they can give you a sign off.

10.4.1 Final presentation

When you have sign off on your analysis from your manager, you should set up a meeting with the stakeholder to deliver your analysis in person. In this meeting you'll want to walk them through each component describing what you did, what you learned, and what you chose not to look into. You will have spent so much time with the data creating the analysis that you should be pretty comfortable explaining it and answering questions.

Depending on the stakeholder you may find yourself being peppered with questions throughout your presentation, or the person may save questions until the end. Questions may range from calm and curious ("why did you use dataset X instead of dataset Y") to critical and concerned ("Why don't these results align with the other team's work? Are there errors in your code?"). How you should handle questions in many ways is the same as getting questions during job interviews: be upfront with what you know and what you don't know. It's okay to say you need to look into something. As much as possible be open with your reasoning ("we used dataset X because it covered the time period we cared about") and when you don't know something ("I am not sure why they don't align with the other team; I'll look into that."). That being said, most of the time these meetings are calm and conflict-free!

No matter how good your analysis is, you will inevitably get a question of the form "well what about _____?" where the blank is something you haven't looked at in your analysis. For example, someone might say, "Well, what about if you only use the last month's data in the analysis?" This is a natural thing for people to do because of the nature of data science — you can always find more ways to slice the data and ideas for what might possibly be useful. This is especially common in situations where the analysis proved inconclusive. In these situations, the person making the request often wants to jump in with a hope of something that might suddenly prove conclusive.

As a data scientist, the best thing you can do in these situations is to try and gently push back against these requests. While they occasionally can prove useful, they can just as easily

end up drawing no new conclusions, all while causing you to lose days of time trying to work through them. As the data scientist, you should have the best knowledge of what would have a chance of being valuable, and if you don't think something would be useful you can lean into that. Often times when doing an analysis, the business question trying to be solved is so abstract that you could never give a truly definitive answer. And just like when you were doing the analysis and had to avoid trying method after method to find a result, after the analysis you have to know when to stop.

10.4.2 Mothballing your work

When the final analysis is delivered and approved, you will quickly be asked to move onto the next set of work, like another analysis. Before you do, however, there are a few small steps that will make your life much easier in future. There is a good chance that at some point down the line, either months or years away, you'll be asked to redo the analysis with more recent data. If you spend some time documenting your work, it'll be much easier to complete that repeat analysis. The steps are

- **Double check that you can rerun the whole analysis**—earlier we discussed making your analysis a one-button run; at this point you should do one last check that it still works.
- **Comment your code:** Since you might not look at your code again for years, even light commenting can help you remember how to use or modify your code.
- **Add a readme:** A readme file is a simple text document covering what the analysis is for, why it was done, and how to run it.
- **Store your code safely:** If you are using git and GitHub you've already done this, but if not consider how someone can access the code far from now.
- **Ensure that the data is stored safely**: Check that any data file is stored in a safe place besides your laptop, like cloud services such as OneDrive, Dropbox, or AWS S3. Data sets stored in databases should also ideally be checked that they won't be deleted.
- **Output is stored in a shared location**: The most common way people share analyses are as email attachments, but that's not a good way to archive them. Place your results somewhere that other team members and people in other parts of the business can access them.

Once that work is all done you can call the analysis truly complete. As you make more and more analyses, you'll find methods and techniques that work best for you and you'll get better and faster at making them.

10.5 Interview with Hilary Parker, a Data Scientist at Stitch Fix

Hilary Parker works at Stitch Fix, an online personal styling service, where she creates machine learning models to help suggest clothing to customers. Previously she was a Senior

Data Analyst at Etsy. She has a PhD in biostatistics from the Johns Hopkins Bloomberg School of Public Health.

10.5.1 How does thinking about others help your analysis?

Just about every analysis I start by trying to understand "who wants what?" For example, was this work requested because the product manager needs to make a decision and they don't feel like they can until they get this analysis of an experiment? Is it that we're trying to drive a strategic vision and in order to get people comfortable with it, we're going to need to show that we believe this would make X dollars over Y years? I make sure to sit down and talk with the eventual consumers of the analysis to understand their context.

When you're presenting, the most important thing is to understand the audience and where they're at and what their goals are. Do they want to understand the nitty gritty or not? What would be most compelling to them? If they seem eager for more information, you can offer more statistical details, but if they're disengaging, you want to scale that back.

10.5.2 How do you structure your analyses?

I think structuring the analysis in a way where it is approachable is important. I make sure that I have a short summary at the top and that I'm not making complicated graphs, since most people can't absorb them quickly. I also don't do a stream of consciousness notebook for an analysis, which I see a lot of people doing in an industry setting. The comments in notebooks look like text so you add more and more comments and deliver that. You end up delivering something that is "here's where I started and here's where I ended." But you actually want to flip that to: "here's the conclusion and in the appendix you see where I started." Keep in mind the idea that a human will read it and what's going to be the easiest to produce quickly may not be the most readable. I focus on the final format so much that it's like part of the process. I don't ever have to translate a big notebook and make it pretty—I'm making the pretty thing the whole time.

10.5.3 What kind of polish do you do in the final version?

I think color themes go a really long way. A lot of companies will have their corporate color theme—Stitch Fix has a color theme in our branding. We have ggplot2 templates that import the colors from our color palette. Stuff like that is really effective since it makes people in the company feel familiar. We have the same thing with Google Slides presentations. There are Google Slides templates that people use because it looks nice when you do it.

I also think "don't overdo it." One of my early projects at Stitch Fix was launching our plus size business line. We had some rapid analysis we needed to do in order to understand if we were sending out the right sizes. I spent so much time building my little system for how we were going to deliver the analysis. I got all excited about developing this reproducible website that would dynamically update at every x hours to show what was changing. But ultimately the people I was working with didn't look at it that much. I got all excited about building the

website instead of checking in with the partner. It's easy to go overboard with the aesthetics of the analysis. Do as much as you need, but not too much.

10.5.4 How do you handle people asking for adjustments to an analysis?

I've been reading a lot about design thinking recently and this happens in a design context all the time. The attitude there is the one that I've taken upon myself: that people are bad at communicating and they're not going to zoom out and think abstractly. In the design world the person is going to come to you and say words about what they want, and you can't take them literally. You have to help them frame the problem. That's part of what a designer's value add is: that they're thinking about the problem holistically and systematically framing in different ways until they come up with the frame that makes sense.

I think that data scientists and statisticians are the same way. Someone's going to ask for some feature because they're trying to articulate unease, and that's one way of expressing it. But you have to figure out what the unease is about. Are you saying that you don't want to make this decision? Is this causing hesitancy? What's going to be the end result of this? Almost any time as a data scientist, you're interacting with one of the consumers. You have to constantly be not just doing exactly what they say, but instead figuring out what this person is actually trying to say. What's the root cause of what they're saying? Is that something that it's even appropriate for an analysis to address? There can be a lot going on and it really pays to have a bird's-eye perspective on the entire situation rather than just iterating until the end of time.

10.6 Summary

- Analyses are documents that highlight conclusions and encapsulate important features of an application of data science to solve a business problem. They are critical for data scientists.
- A great analysis requires understanding the business problem and how data can solve it.
- When making the analysis, constantly think of the end goal, use simple methods with clear visualizations, and be ready to share your work.
- Managing the process of creating the analysis is important to keep the work focused on the goal and to ensure it has a clear end.

11

Deploying a model into production

This chapter covers

- Building a machine learning model to use in production
- What APIs are and how they're helpful
- Deploying a machine learning model

This chapter is written to cover the essential concepts of a machine learning engineer's job—a person who creates machine learning models and deploys them for the business to use. If your work instead involves creating analyses and reports, it's easy to be scared of this material. Don't be! The gap between decision scientist and machine learning engineer is smaller than it seems, and this chapter will be a helpful introduction to the concepts.

Sometimes the point of a data science project isn't to answer a question with data—it's to create a tool that uses a machine learning model to do something useful. While you could do an analysis to understand what items people tend to buy together, it's a different task to make a program that recommends the best item to show a customer on the website. The work of taking a machine learning model and making it so that it can be used by other parts of the business, such as on the website or in the call center, tends to be complex and involve data scientists, software engineers, and product managers. In this chapter we'll discuss how to think about how to make models that are part of a product, and how to get them off your laptop and into a place where they can function.

Two minor notes before we dive into this topic. First, because the task of creating code that runs in production is fairly technical, this chapter is more technical than the others. Since we want these topics to be easy to understand for people who have less familiarity with the concepts of software development, we'll focus more on the concepts and ideas than the technical specifics. Second, because we are focusing more on the concepts, at times we will

make general statements that may not always be 100% true. This is an intentional decision to help with readability. If you are already familiar with these topics and can think of a counterexample to something we've written—you're probably right!

11.1 What is deploying to production anyway?

When people say "deploying into production" what they mean is taking code and putting it on some sort of system that allows it to be continuously run—typically as part of a customer facing product. Deploying is a verb—it's the act of moving it to a different system—and production is a noun—it's the place where code that is part of a product runs. Code that is in production needs to be able to work with minimal errors or issues since if the code stops working, customers notice.

While software developers have been putting code into production for decades, it's becoming more and more common for data scientists, specifically machine learning engineers, to train a machine learning model and put it into production as well. Training a machine learning model to be put in production is similar to training a model as part of an analysis, but there are substantially more steps after the model is trained to have it ready for production. Often the act of building the model for production starts with an analysis — first you need to understand the data and get business buy-in; then you can think about deploying to production. Thus, the two acts are quite intertwined.

To get a better sense of what deploying to production means, here is a quick example. Suppose that a business stakeholder in a company thinks that too many customers are leaving and asks a data scientist to do an analysis of customer churn. As part of the analysis, the data scientist builds a model and shows that there are several key indicators of churn. The business stakeholder loves the analysis and realizes that if customer care agents working in the call center knew the customers likely to churn, the agents could offer discounts to try and keep them from leaving.

At this point, the data scientist needs to put the model in production. The model that lived on the data scientist's laptop must somehow run every time a customer calls support. On the laptop, the model took a few minutes to assess many customers at once, but in production this model has to run on a single customer the moment they call in, pulling customer data in from other parts of the company and using that for a rating.

Most production machine learning models are similar to this — they need to work in near-real time to make a prediction or classify something based on provided data. Famous examples are Netflix's model that predicts which movies a person would like, Facebook's facial recognition model which takes an image, finds faces in it, and matches those faces to identities, or Google's Gmail auto-complete models, which takes text as you're writing it and predicts the next word.

There are substantial steps that models used for production need to go through. First, the models need to be coded to handle any possible scenario that might happen when the code is live, so they are less prone to errors. When doing an analysis, small amounts of weird data

can be filtered out and not fed into the model without disrupting the analysis results. In a production model, the code needs to run regardless of how strange the input data is. For instance, it's probably fine for analysis purposes if a natural language processing model crashes when run on an emoji since you can just ignore input data with emojis. For a production model, if the code breaks when an emoji shows up, that could cause the product the machine learning model supports to break too - imagine if the Gmail webpage crashed every time you typed an emoji. The production models either need to be made to handle weird cases themselves, or there needs to be code that fixes special weird cases before they ever get to the model.

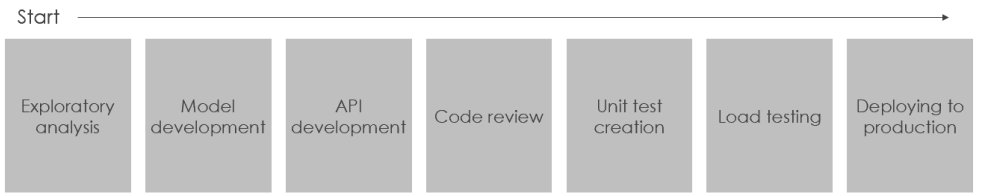


Figure 11.1: Example process of creating a production machine learning product

Production models must also be maintainable. Because they are being continuously used in products, they occasionally have to be retrained on newer data—or coded in such a way that they automatically retrain themselves. They need ways of monitoring how well they are working, so that people in the company can tell if they are no longer working as well, or if they suddenly stop working altogether. And since they will possibly be running for years, they need to be coded in such a way that they follow standards of other models and can be updated over time. Coding one model in a long-dead programming language that few people know is bad for an analysis and catastrophic for a production model. Check out Figure 11.1 to see an example process of creating and deploying a production machine learning model.

The rest of this chapter will cover three concepts: how to create a machine learning model that is suitable for production, how to deploy it into production, and how to keep it running over time.

Production for different types of data scientists

How much you will have to think about production systems varies greatly by the different types of data scientist.

Machine learning engineer – This is pretty much your whole job. By the time you're comfortable in your role as a machine learning engineer you should be comfortable with everything discussed in this chapter.

Analyst– an analyst may have to deal with production systems depending on the complexity of their reporting. If the analytics team is creating consistent reporting on a recurring basis, then it's best to productionize the reporting

systems. By making it so that the reports refresh themselves automatically, the analytics team is free to do other work. This is especially true for dashboards where the expectation is the systems are in production updating themselves.

Decision scientist – because the work of a decision scientist is mostly ad hoc, there aren't as many opportunities to create production systems. However, if models created by decision scientists are handed off to machine learning engineers to productionize, having a better understanding of production systems will help. Decision scientists also can create interactive tools for the business from libraries like Shiny or Dash that need to be deployed and maintained in production systems.

11.2 Making the production system

A production system based around a machine learning model starts with the same steps as making a machine learning model as part of an analysis—you need to find the appropriate data, do feature selection, train the model, and get business buy-in. After those steps are complete, a fair amount of additional work has to be done. First, the model needs to be converted into a format that other programs can utilize it. Typically that's done by creating code that allows the model to be accessed from other systems in the company, just like a website (this is usually in the form of an API). Then, the model has to have code added to handle many possible cases to ensure it has as little downtime as possible. This includes adding tests to the model to ensure that it's correctly handling all of the data it could be expected to process. The model is deployed to a test environment to ensure it functions correctly and that the API can handle the amount of traffic that will be hitting it once it's live. Once all these steps are complete, the model is finally deployed into a production environment.

11.2.1 Data collection

When collecting data to train a model for an analysis, you need to find a suitable set of historical data that contains a good signal within it. While this is also necessary for a production model, it's often not enough, since the real-time component of the model needs to be considered. Take the example from earlier, where a company needs a model in production to predict in real time if a customer is going to leave or not. If a customer churn model was going to be used for an analysis, a historical collection of customer attributes (number of purchases, years since first purchase) that was collected a few months ago would be great for a model. Because the production model needs to be able to predict in real time if a customer will leave, code will need to be written to somehow find out what the customer's attributes are at the time the model is being called upon. If customer #25194 is calling customer support, the code for the model needs to know at that precise moment how many orders the customer has made, so the model can make its determination.

The difference between using historical data to train a model and feeding the model real-time data when running it can often be drastic. For technical reasons regarding how the data is collected, there may be a lag of hours or days before data is put into a database or storage location that a data scientist can access. Alternatively, there can be situations where the data

is available in real time, but the values aren't stored historically. For example, there may be a way to query if a customer is located internationally at the moment, but the data on if the customer was ever international was not stored.

When looking for data to create a model for production, consider what data will be needed in real time when the model is run. Will the data be suitably fresh for use? Can it be accessed via a database connection or some other method that someone can set up? It is not uncommon for machine learning projects to fail because of dataset issues.

11.2.2 Building the model

Once you have a suitable dataset, you can start building a machine learning model. This is a broad topic; if you want to learn about how to build a machine learning model, from feature engineering to model training and validation, there are lots of books and internet resources available to cover this topic. We include some recommended resources in the appendix of this book. That being said, when making a model specifically for production there are a few things you want to keep in mind.

YOU HAVE TO PAY EXTRA ATTENTION TO HOW THE MODEL PERFORMS IN ALL CASES

Because your model is going to be relied on by other systems to work effectively on whatever data those systems choose to pass to it, you need to understand how the model will work in all cases. For example, suppose you were making a machine learning model as part of an analysis to understand what products customers are interested in, e.g. a model that predicts the product a customer would buy next. If the model you built correctly predicted 99% of the purchases, but 1% of the time predicted that the customer would order Nicolas Cage temporary tattoos, the model would be a huge success — by understanding the vast majority of customers, you could help the business make an informed marketing decision. If on the other hand you were going to deploy that model to show recommended products on the company website, then that 1% could be catastrophic and cause you to lose customers (or at least, those customers that don't appreciate Nicolas Cage's unmatched charisma). In this way, what happens on the margins is really important in production systems but not in analyses.

YOU WANT THE MODEL TO BE SIMPLE

Once the model is deployed into production and running so that customers interact with it, you'll inevitably run into a situation where the model does something strange and you want to understand why. For instance, if it's a customer churn model, it could predict every customer in Alaska is going to churn for some unknown reason. Or a recommendation engine on an outdoor products website might only recommend kayaks. At this point you'll need to dig in to try and figure out what is going on and if the model needs to be changed in some way.

If you use a simple model like a linear regression, it should be fairly straightforward to trace the computation done to make the prediction. If you use a complex method like an ensemble or boosted model, it becomes much more difficult to understand what is going on.

While you should use a model that's complex enough to solve your problem, as much as possible you should use the simplest acceptable model, even at the cost of accuracy.

One interesting real-world story about this involves the Netflix Prize. Netflix ran a contest to see if a team could create an algorithm that improved their movie recommendation results by 10%, and in 2009 they awarded a $1 million prize for it. As covered in a Wired article, Netflix actually never ended up using the winning algorithm. The algorithm was an ensemble method that combined many different models, and the engineering complexity of running and debugging it was so high it wasn't worth the increase in accuracy. Despite Netflix paying a high price to have an accurate model, they realized there were more important things than accuracy. If Netflix with its army of data scientists and engineers doesn't maintain a highly complex model, it's hard to say that many companies should.

11.2.3 Serving models with APIs

As of the time of the writing of this book, most machine learning models are served as APIs — Application Programming Interfaces. This means the machine learning model code can run on one computer system (also called a service), and other systems can connect to it when they need the model to run on their data. For example, if a company has one system that runs the shopping website and wants to add a machine learning model to give a discount if the customer is predicted to unsubscribe, rather than trying to get the machine learning code inside the website code, they could set up a second system that has the machine learning model and the website could periodically connect to it.

The concept of breaking up different parts of a system into small microservices is all the rage in software engineering and is covered in beautiful depth in many different books. As data scientists, the important concept is that we want to set up one computer system that solely runs the model so that other systems can use it.

Modern APIs are done using web services—colloquially called REST APIs. A REST API is basically a tiny website, but instead of the website returning HTML to render in a browser like most websites, it instead returns data, usually as formatted text. These requests use the HTTP protocol, which is the same protocol web browsers use (and which is why websites start with http or https). For instance, a weather API could be set up so that if you go to the URL http://exampleweather.com/seattle_temperature it would return "45." For a machine learning model API, you would want to go to a particular website and get a prediction. In the case of a machine learning model, somewhere in the company network there could be a site that predicts if a customer would leave, like http://internalcompany.com/predict?customer=1234 would return a number between zero and one representing the probability of that customer leaving.

Designing the API involves making decisions like what URLs return what data and what types of requests to use. Making the design understandable to users is important to actually having people use it, so often as much care has to be put into thinking through the interface as in the creation of the model.

Having a machine learning model run as an API web service is great for a bunch of reasons. First, because it's a web API, anything in the company can use the model — including both live systems and other data scientists running analyses. The same churn prediction model being used on the website can be queried for an analysis by a person doing decision science. Second, because it works like a website, almost any modern technology can connect to it regardless of technical platform. If the model is written in R, it can still be used by a website written in Node.js, or by another analyst using Python. And because it's hosted as its own website, if for some reason the model stops working, it's less likely to take other products down with it. If the company shopping site uses the model and suddenly can't connect, the shopping site should still be running.

11.2.4 Building an API

APIs are great for machine learning models, but some additional coding is needed for them. Both R and Python have packages that do this for you—Plumber and Flask respectively. When you run an R or Python script with these packages, it will take your function and expose it to an endpoint on your computer. You could specify that going to the URL "http://yourwebsite.com/predict" runs your machine learning R or Python function, and then returns whatever the function result is. You could then go into a web browser to call your code! Assuming you're running this code on a laptop or desktop computer, if you set your computer to allow outside traffic by adjusting the firewall, then other people could hit your API! But the moment you stop running the API hosting program (R or Python), no one will be able to run your model.

While both R and Python make it easy to serve a model as an API, there are design decisions that need to be made. First, what data is going to need to be passed as input to the model in an API request? For instance, suppose you were making a model that would predict a customer's likelihood to churn based on their tenure with the company, the amount they've spent with the company, and the number of times they've called customer care. One possible API design would be for a person to do a request with the customer's unique ID in the URL. For example, to look up the customer with ID 1234 you could go to: http://yourwebsite.com/predict?customer_id=1234. Another option would be for the user to have to first look up all the customer information themselves, and then include that as the body of a request. So for a customer with a tenure of 1.7 years, a total spend of $1257, and three calls to the contact center, you could send a request to http://yourwebsite.com/predict, and have it have a body of {"tenure":1.7,"spend":1257,"calls":3}.

These are both valid options for an API design, but one requires your API to do all the work of looking up the customer details, while the other makes the API user have to look up the details. It's generally not a good idea to make these decisions alone—the more you can loop in the people who might use your API, the more likely they are to be happy with it.

Once you've designed your API, go talk to the people who are going to use the API and show them how it works. They should be able to give you feedback on your design. Ideally you should also share documentation on the API with them.

On plumber – an R package for serving R code as a web API

Jeff Allen works at RStudio and is the creator of the R package plumber. Plumber allows people to create web APIs in R so that machine learning models can be used across an organization.

Plumber wasn't started from a company deciding to pour lots of engineers and funding into creating it—it had more humble origins. In 2012 I was working in a Biostatistics group at a research center. This group used R for much of their analysis and had brought me in to help them create better software. Primarily, we had three different audiences:

a. Other biostatisticians who use R and wanted to leverage or evaluate the methods we had developed.
b. Non-technical users, such as clinicians, who just need the results of our analysis to answer questions like, "which drug would be most effective for this patient?"
c. Technical users who want to leverage our analysis but aren't interested in or don't have the computational resources to run our R code.

The first audience was the easiest; R comes with a robust packaging system that allows you to bundle your code and data in packages that can be shared and used by others. The second audience was a bit harder to serve at the time. Today, Shiny is the obvious solution to this problem and offers a convenient way for R users to continue working in R to build rich, interactive web applications that can be consumed by a non-technical audience.

It was this third audience that remained difficult to address. Some users already had an existing application written in another language like Java but wanted to invoke some of our R functions from their service. Others had a simple, automated pipeline and wanted to leverage some computationally intensive R function that we had defined. In all of these cases, what they really wanted was something that they could call remotely and rely on us to internally do all of the R processing and then send them the result. In short, they wanted a remote API for R.

It was years later that I actually had the opportunity to start working on the package that would become Plumber, but I carried this motivation with me. There is a group of people in most organizations who don't know R but would benefit from the analysis that their data scientists are creating in R. For many, a programmatic and structured interface is required, and web APIs offer an elegant solution. Thankfully, the authors of the Shiny package had already solved all the hard problems around creating a performant web server that could be used by R packages to service HTTP requests. The remaining piece was to create an interface by which users could define the structure and behavior of their API.

My hope is that Plumber offers a solution to that technical audience, so that these programmatic users can benefit from R just as effectively as the other aforementioned audiences have. Watching Plumber's growth over the years, both in features and usage, I think it was able to strike a chord with R users. Because R can be conveniently surfaced over an API, it's now earned a seat alongside other programming languages that are more familiar to traditional IT organizations. It's been fun watching people use Plumber to accomplish things that I could never have done.

11.2.5 Documentation

Once you have a working API, it's a great time to write documentation around it. The earlier in the process you have documentation, the easier it is to keep it maintained over time. In fact, it's a great practice to write your documentation about your API before you write the first line of code. In that case, the documentation is your blueprint for creating the API, and people who will be using your model will have plenty of time to prepare for it.

The core part of API documentation is the specification for the API requests: what data can be sent to what endpoints, and what is expected back? This documentation allows other

people to write the code that will call the API and know what to expect. This documentation should include lots of details like:

- The endpoint URLs: (http://www.companywebsite.com/example)
- The request types for each URL
- What needs to be included in the request
- The format and content of the response

This documentation could live in any text document, but there are also standard templates to store it, such as OpenAPI documents. An OpenAPI document is a specification for how to write API specification files that can easily be understood by users or computer systems.

Ideally you won't be the person keeping the API running forever, so you'll also want documentation around what requirements the API has to run on your system and how to install it somewhere else. This will allow the person who takes over to get the code working themselves and make changes as needed.

Finally, you'll want to have some documentation around why the model exists and the basic methods behind it. This is useful for cases when you are no longer working on the product and knowledge on why it was created is lost.

11.2.6 Testing

Before a machine learning model is put into production and customers rely on it, it's important to make sure it works. When a machine learning model is trained, part of the process is checking the model output and ensuring accuracy, which is helpful but not fully sufficient to know whether the model will work. Rather, the model needs to be tested to check that it can handle any input that it could receive without failing. If a churn model is being put into production that has an API that takes a numeric customer ID as input, what happens if the customer ID is blank? Or a negative number? Or the word "customer?" If in these cases the API returns a response that the users don't expect it can be bad. If the bad input causes the API to *crash it could be catastrophic. Thus, the more these issues can be caught in advance the better.*

There are many different types of testing, but for the purposes of creating a production machine learning model one particularly important type is unit testing. Unit testing is the process of testing each small component of the code to ensure the system will work in practice. In the case of a machine learning API, that often means testing that each endpoint of the API behaves as expected under different conditions. This can include receiving as input enormously large numbers, negative numbers, or strings with weird words in them. Each scenario is turned into a test. For example, for a machine learning model that classifies text as either positive or negative sentiment, the test could be, "on input 'I love you,' we expect the API response to be 'positive'." Another test could be that if the input is a number like 27.5, the code returns a `could-not-compute' result rather than crashing.

Beyond testing the API endpoints, functions within the code can be individually tested too. The goal is to have 100% coverage, meaning every line of code in the API has been tested to

function correctly. Each time the model is going to be deployed, the tests are checked and, if any of the tests fail, the issues have to be resolved.

It's easy to brush off testing as something there isn't time for, but it's often the only way major issues are caught before being put in front of customers. Writing a bunch of checks feels like busywork compared to the task of building a machine learning model, but it's extremely important and shouldn't be ignored.

11.2.7 Deploying an API

If you have a machine learning model coded so you can run it on your laptop, it's not much work to turn it into an API that runs on your laptop. Unfortunately, sometimes you want to turn your laptop off or use it for watching Netflix, so having an API continuously running on it isn't a long-term strategy. To have your API running continuously in a stable way, it needs to live on a server somewhere so that it'll always run. The process of moving the code to a server to run is what we mean by "deploying it." Setting up the server to have the code always running is a bit more work beyond creating the API.

The term "server"

When you hear "server" it can sound intimidating, like some special computer that normal people don't understand. In practice, a server is really just an ordinary computer like a laptop, but it runs somewhere without a screen. Instead of walking up to a service and logging in, people connect remotely with other computers, which simulates being in front of it. Servers run almost the same operating systems as computer laptops—Windows or Linux with minor tweaks. If you connect remotely to a server it should look very familiar, with the same start menu for Windows or a terminal for Linux. The advantage of a server is that people usually leave them running and they tend to live in places safer than a home office. But there is no reason you couldn't take an old home PC, put it in your closet, then treat it as a server—many software engineering hobbyists do just this.

When people talk about using cloud services like Amazon Web Services (AWS), Azure, and Google Cloud Platform (GCP), what they mean is that they're using servers that are rented from Amazon, Microsoft, or Google. But just because you're paying a big company for them doesn't mean they're different computers. You can just think of these as "expensive sky laptops."

There are two basic ways to deploy an API to a server: you can run it on a virtual machine, or you can put it in a container.

DEPLOYING TO A VIRTUAL MACHINE

Enterprise servers are generally machines that are both extremely powerful and extremely expensive. It rarely makes sense to have this powerful machine devoted to a single task because for most tasks it would be overkill. Instead, the server will be doing many tasks at once, but it would be catastrophic for one task to crash the computer and then cause unrelated tasks to crash too. Virtual machines are a solution to this—they are simulations of computers. The large expensive computer will run many simulations of other computers at the

same time. If one simulation crashes it's no problem; the other ones keep working. A virtual machine can, in almost all cases, be treated exactly the same as a regular computer—if you log into one you couldn't tell it was a virtual machine unless you were looking for it. Every time you use AWS, Azure, or GCP to access a computer, you're connecting to a virtual machine. If you ask your IT department to get a server for you, they'll likely make you an on-premise (non-cloud) virtual machine as well.

Virtual machines are great since they're simulations: you can easily turn them on or off. You can also take snapshots so that you can go back to an earlier version of it, or have many copies running at once. You can share a snapshot with someone else and they can run it too. You can also close your eyes and pretend it's a regular old laptop and generally be fine, assuming you use your laptop with your eyes closed.

Because it's a regular computer, the simplest way to deploy your code to the machine is to install R or Python, install the libraries you need, copy your code onto it, then hit run. These are the same steps you'll use to get your API running on your laptop! If you want to make changes to the API, just copy the newer version of the code to the virtual machine then hit run. It really is the case that you have a system in production using only the steps: (1) start a virtual machine, (2) install the programs and code you need to run the machine learning model API, and (3) start running your API. Given how much people talk about the complexity of making production systems it's shocking how easily you can do it in a barebones way.

One major hassle with this simple method of copying and pasting code onto a virtual machine and hitting run is that you have to manually move the code every time you make a change. This is both laborious and also prone to error. It's easy to forget to move the code over or lose track of which version is on the virtual machine.

Continuous Integration (CI) is the practice of having code be recompiled automatically every time it is committed into a repository. CI tools can monitor git repositories and see when changes are made, then rebuild the software based on that. If you're using R or Python there isn't likely to be recompiling necessary, but the build process can run steps like rerunning the unit tests. *Continuous Deployment* (CD) is the practice of taking the output of continuous integration tools and automatically deploying it in production systems. CI/CD refers to using both of these together.

Thus, a CI/CD tool will check your repository for changes and, if there are some, will run your build process (like check unit tests), then move the resulting code onto a virtual machine. As a data scientist you don't have to worry about making changes to the virtual machine, the CI/CD tool will do that for you. Setting up a CI/CD tool yourself isn't an easy task, but if your company has a software development team it is likely they already have these tools set up and you can use them.

Another improvement to using virtual machines is running multiple virtual machines at once. If you expect that your API is going to get a lot of traffic, you can make copies of the virtual machine then run them all at once and assign traffic randomly to a machine. Further, you can monitor how active each virtual machine is and start and stop extra copies of the machine as needed. This is called auto-scaling. While practical for large systems it's also a bit

of a pain to set up, and if you are in a situation where you need it you also are likely in a situation where there are software developers around to can help.

DEPLOYING TO A DOCKER CONTAINER

Setting up and running a virtual machine can be quite a hassle. Because each one is a simulation of a computer, setting it up is as annoying as setting up a regular computer. You have to install each program, change each driver, and get it configured just right. It's really difficult to document all of the steps needed, and if someone else repeats the process it easy to make mistakes. Another problem is that since each virtual machine is a simulation of a computer, they take up a lot of space since they have to contain everything a regular computer does.

Docker is a solution to these problems. To use a metaphor from Mike Coleman on the Docker blog: if a server full of virtual machines is a neighborhood of houses, then docker containers are a set of apartments in a single building. While each apartment is a fully livable unit, they share services like a hot water heater. Compared to virtual machines they are much easier to set up and more efficient to run.

Docker allows you to easily specify how a machine is set up, and by having a shared specification across distinct machines you can share resources. This allows the creation and maintenance of production systems to be far easier than with virtual machines, which is why they took the software development world by storm.

There are three concepts that are important to understand Docker. First, a *dockerfile* is a text file that contains all the steps to set up the simulated machine. These can include steps like "Install Python 3" or "Copy over the saved model file to the machine." Most of the steps are the exact same as Linux bash commands, so if you're used to reading those then a dockerfile should feel familiar. When you tell Docker to build a dockerfile, it follows the steps to create the machine and saves the result as an *image*. Thus, an image is a stopped snapshot of a computer. Finally, a *container* takes an image and starts it — a container is a virtual machine.

Docker has lots of advantages over the traditional deployment methods, but using Docker to deploy a machine learning model into production will likely only work if there are other people in the organization using Docker containers. If that's the case, the company will have someone who knows how to create a Docker container, deploy it, and monitor it to see if it's continuously working. If not, there'll likely be pushback at having the machine learning models deployed in a non-standard way.

Because they are more complex to get started with compared to virtual machines, if you've never deployed code before it may be easier to get started with virtual machines first. Even if you can't use Docker for deploying models, there are lots of benefits to using Docker for reproducible analyses. It's definitely worth it to at least gain a little experience in the tool at some point, even if it's not immediately clear that you have a career need for it.

11.2.8 Load testing

If lots of systems are going to be using your model, or one system is going to be using your model a lot of times at once, you'll want to make sure the API won't fail under the stress. This could happen because the system it's running on runs out of memory, because the system takes too long to process each request and has a longer and longer queue, or any other number of terrible things.

The easiest way to make sure this doesn't happen is to run a load test—a test where you make a high number of requests to the API at the same time and see how it does. Usually you run with a number of requests that is at least twice as much as you could ever expect to see. If the API gracefully handles these requests, then you're golden. If it crashes though, you know you need to change your code to be more efficient, scale your system up, or make other changes.

11.3 Keeping the system running

Even after your API is successfully deployed and being used, it's not enough (it's never enough). Either you or someone else in the organization will have the job of continuing to ensure it works. Some companies have a Development Operations (DevOps) team whose charter is to make sure the APIs are always working. Even if the API is working fine you still may want to make adjustments for other reasons. Here are three important considerations for your API maintenance over time.

11.3.1 Monitoring the system

It's a good idea to continuously monitor how the model is doing. How many requests is it receiving each hour? Are the model predictions still accurate? Are there any errors happening? The easiest way to keep track of these metrics is by having your API include logging and telemetry. Logging is recording data on internal issues within the tool, for instance every time the model has an error. Telemetry is recording events that occur, for instance every time a request or particular prediction is made. Additionally, alerting can be set up for when issues occur.

Logging can be as simple as having your API write information to a file every time an event occurs. Then to check the logs you can just enter the Docker container or virtual machine. Telemetry generally involves sending the event information to a remote location (such as a centralized server), so that the telemetry of many different systems is all located in one place. A dashboard can then be made so that the telemetry can be viewed and monitored in real time.

Alerting tools are used so that when something is going wrong people in the company find out. These can be automatic emails or slack messages that are sent when a specific set of events occur. For instance, if the model API has a telemetry event for when a request is

received and no requests happen over an entire day, then an alert email could be sent out to let someone know that the system isn't receiving traffic but probably should be.

These different monitoring systems are often used in conjunction, and companies try to standardize so that all of the company APIs can be monitored in the same way. Like much of this chapter, the more you can work with the standards of your organization the more useful your tool will be.

11.3.2 Retraining the model

It is often the case that at some point after a machine learning model is released into production it will start to perform less well. Machine learning models are trained on data, and as time passes that data becomes less relevant. A machine learning model to predict churn may falter as customers from new regions start to engage with the company. When the model is sufficiently non-performant, the model will need to be retrained.

The simplest solution to retraining is that when the model does badly, repeat the steps you followed to train the model in the first place but load a newer version of the data. This likely means loading data in R or Python onto your computer, rerunning the scripts, then putting it into production in the same way. This is nice because if you did it once you can do it again. And, in fact, plenty of important production machine learning systems are handled in this way at large impressive corporations.

One thing you can do to make this more sophisticated is to create some standard schedule for the work to be done. Instead of trying to keep an eye on the model metrics and use your gut to decide when to retrain it, set a standard practice of doing it every *n* weeks or months. It takes the guess work out of choosing when to do an important act.

More critically, having the retraining be done on a standard schedule means you can automate the process. If you have a Python or R script that loads the data, builds the model, then saves it somewhere, you can set up a system to automatically do that on schedule. In fact, this retrainer system can be put into production itself, so you don't have to spend time doing the work. Such a system can also test that the new retrained model is performing just as well or better than the previous one, and if not, send out an alert to the data scientists. These sorts of more advanced retraining processes are becoming more common and cloud tools like AWS SageMaker have support for them.

Automatic retraining pipelines are sophisticated and in vogue, but at the end of the day so long as you are retraining your model at all you're doing great. Data scientists get into trouble when they build a model, deploy it into production, and stop paying attention to it as the model does less and less well over time. By not continuing to monitor performance of a model and fix it if necessary, you run a real risk of doing damage with your work instead of helping. Keep an eye on it.

11.3.3 Making changes

If your model in production is successful for the business, you'll inevitably want to make changes to the model to improve it. You may want to pull in more data sets or change the machine learning method to improve the performance of the API. You'll also hear from people around the business about features that people want in the model or issues people find with it.

Just as in the previous chapter about analyses, these sorts of one-more-thing changes can present real issues. It's not clear that doing this work is necessarily worth the time, even if it's interesting or seems important to someone. If it takes three months to get the model from 84% to 86% accuracy, you'll lose three months that you could have spent on something else. Or a feature that seems important to a particular stakeholder may not actually affect many customers. A successful machine learning model in production will draw the attention of many people, and as the data scientist who helped create it you should try to ensure time spent improving it is well-spent.

11.4 Wrapping up

This chapter covered lots of different concepts, some of which you may have already been familiar with and some not. While not all of the topics may be relevant to your work, it's great to have a basic understanding of what they are. There are lots of good resources in books and online to learn more about these topics—especially since there is so much overlap with software engineering. As data science continues to change as a field, these topics will stay important and are worth continuing to learn.

11.5 Interview with Heather Nolis, a Machine Learning Engineer at T-Mobile

Heather Nolis is a machine learning engineer on the AI @ T-Mobile team, where she helps put R and Python models into production that are hit millions of times a week. She holds a master's degree in Computer Science and a bachelor's in Neuroscience and French.

11.5.1 What does "machine learning engineer" mean on your team?

I take the models that data scientists make and engineer them into products that the team maintains. For a long time, T-Mobile had data scientists who would sit around building beautiful models and doing cool analyses that they would then send over to the software engineering department to put it into production. The idea was that this would let the work have a real business impact, but it was really difficult for the engineers to actually use the work because there was a huge language barrier between data scientists and engineers. My goal is to sit in the middle so I can understand all of the things that went into an analysis or are important for a given model and then communicate that with engineers.

11.5.2 What was it like to deploy your first piece of code?

My first deployment was my first week ever as a software developer. It was on an already existing product, and at first, I didn't understand why it was risky. When you're coding on a computer, it's fine if you run the code 50 times to test it. But in production if one piece of code has a bug, it can cause a huge problem for your company, so it's not just you on your laptop that you're inconveniencing by running code that doesn't work. In my first release when I thought I was done, I actually had to do three hours of integration testing before it was allowed to release.

11.5.3 If you have things go wrong in production, what happens?

Early on I built a Twitter-based tool that would recommend you to the nearest T-Mobile store on social media. I wrote it in Node.js, which our team didn't support at all, but I figured: "I will solve it for me and show people that it could be done and then somebody more qualified can engineer it." That's where I learned that "somebody more qualified can engineer it" never actually happens—my code was what was put into production.

We released it and, me being brand new to Node, it wasn't beautiful code by any means. It worked and was secure but having limited production experience I lacked confidence that it would run in production. Other engineers were nervous because it was a language that we didn't yet support. For the next two months, I got called in every single time any service had a hiccup. I had to be there because I took a risk on releasing something in a different language on a new platform—people assumed that any time anything weird was from me.

Every single time that I got notified for that entire two months, I felt like it was my problem. But was never actually my code that broke in production! I think that's something to remember: when you put things in a production for the first time, there's a chance that you do break everything, but other people's things don't work too. It's not just your code. You don't have to be scared when things break.

I will also say this about putting things in production: of course, I would always like to build the nicest models with all the coolest bells and whistles, but that doesn't always help build the product we want. In the end, we have to sacrifice many of these things to have robust code that will continue to function. My job as a machine learning engineer is to understand these tradeoffs and drive the creation of a product.

11.5.4 What's your final piece of advice for data scientists working with engineers?

Two keys to working well together are to understand their language and appreciate what they care about. For the understanding their language, consider that things which might seem like a normal sentence to you sound to a machine learning engineer like they walked into your house and you had a big old console CRT TVs–they're wondering: "Did I step back in time?" My favorite example of that was when we got our first data scientist on the AI team at T-Mobile. At one point I told the data scientist, "Can't we just put the R model into production as

an API?" She asked, "You want me to run R as a web server?" I had to like step back for a second because when I hear the words "web server" I hear: "hello, I'm coming from the 1980s!" I was completely turned off by that even though she meant exactly the same thing.

For the latter, at the end of the day, data scientists feel good about their jobs when they're building accurate models. What makes me, an engineer, feel good is putting stuff into production so other people can touch it. The only thing that I really value is working code. If you could go to engineer and say I designed an API and I created document that specifies all the inputs and outputs, that shows engineers that your thinking about the problems that they have too.

11.6 Summary

- Deploying to production is the practice of making models continuously run.
- Putting a model in a REST API allows it to be used by other systems.
- APIs can be deployed to virtual machines or as docker containers.
- Look closely into how your company manages code, testing, and deployments of production systems

12

Working with stakeholders

This chapter covers

- Working with different types of stakeholders
- Engaging with people outside of the data science team
- Listening so that your work gets best utilized

It seems like the job of a data scientist would be primarily about data, but much of the work revolves around people. Data scientists spend hours listening to people in their company talk about the problems they have and how data might solve them. Data scientists have to present their work to people so they can use the knowledge gained from an analysis or can trust a machine learning model. And when situations come up like projects being delayed or data not being available, it requires conversations with people to figure out what the next step should be.

Karl Weigers and Joy Betty define the term *stakeholder* in Software Requirements as: *a person, group, or organization that is actively involved in a project, is affected by its outcome, or can influence its outcome.* For a data scientist, stakeholders can be the business people working in marketing, product development, or other areas of the business who use data science to make decisions. Stakeholders can also be people in engineering who rely on machine learning models by data scientist to power their software or make sure the data is collected properly. In some situations, stakeholders can be high level executives. Stakeholders can come from across the company, and different stakeholders have different behaviors and needs.

In this chapter we'll walk through what to expect from the different types of stakeholders you'll encounter during a data science project. We'll then cover how to effectively work with stakeholders and how you should think through communication with people outside the data

science team. Lastly, we'll cover the process of prioritizing the work that stakeholders give you.

12.1 Types of stakeholders

Each of the stakeholders you may encounter during a data science project have their own backgrounds and motivations. While a stakeholder can be pretty much anyone depending on how weird your project is, most stakeholders fall into one of four categories: business, engineering, leadership, and your manager.

Types of stakeholders covered in this section

12.1.1 Business stakeholders

A business stakeholder is a person who is from a department like Marketing, Customer Care, or Product who oversees business decisions. These are people who request analyses to help them make better decisions or machine learning models to increase efficiencies. People in these roles have varied backgrounds: a person in marketing could have an MBA and come from working at an ad agency, while a manager in the customer care department could have started as a care agent with a community college degree and worked their way up. These varied paths to the job will give each person a different perspective when they work with you.

Usually a business stakeholder has little technical background. They might be capable in Microsoft Excel, but with a few exceptions that is the extent of their analytical expertise. Most business stakeholders won't know how to use R or Python or the merits of different machine learning models. But if they have picked up any news article in the past decade, they've heard over and over the value of data when making decisions and how important data science is. Thus, the business stakeholder is in a tricky situation: they have to rely on data scientists to provide them the critical information to make decisions or release machine learning tools, but without technical expertise themselves they have to trust that what the data scientist is saying is right.

Usually a business stakeholder is highly engaged in a data science project. They're there to help kick it off and define the objective for the project. They are there during the project to look at intermediate results and give feedback from the business. And they are there at the end of the project when either the final analysis is delivered or the model is deployed. Since they are the ones who ensure the business gets value from the data science work they have to be constantly involved.

As a data scientist, it's your job to deliver to them what the need for their part of the business to function. That might be an analysis, a dashboard, or on occasion a machine learning model. Not only do you have to do this work for them, but you need to make sure they understand and trust it. If you give them a table with complex statistics and no explanation, they won't understand it and thus won't be able to use it. By being a trusted business partner to them you're enabling them to use data, and they're giving you avenues for more data science to be done within the organization.

The most difficult situations with business stakeholders tend to be when they don't accept the data science results. Situations like when a data scientist makes an analysis and the stakeholder responds with "oh that can't be right." When the facts and premises of the data science work are called into question, there is the possibility of the stakeholder cutting the data scientists out of the loop. In these situations, the best thing you can do is help them understand you did and how you did it. Usually a lack of belief comes from a lack of understanding—and by having a discussion about how things were done it may turn out that you need to change the assumptions in the analysis.

12.1.2 Engineering stakeholders

The engineering teams are in charge of maintaining the code (and potentially physical products) that the company delivers—and when those products require machine learning algorithms or data science analyses they become stakeholders. In some ways, engineers are easier to work with than other types of stakeholders because culturally they have many similarities to data scientists. Just like data scientists, they have technical backgrounds that they gained from schools, bootcamps, or online coursework.

While engineers have extensive technical backgrounds, they often have little experience with key parts of the data science job. While a software engineer writes code, they usually do it with an extremely specific task in mind, like creating an API that queries a specific database.

The job of a software developer does not have the exploratory component of a data scientists, so the idea of spending weeks trying to understand data and gain insights is foreign.

Engineers tend to collaborate with data scientists when a machine learning model is needed as part of an engineering project. This is most often done as a machine learning model being converted into an API in production that the engineers will use for their work (like what was described in Chapter 11). The engineers are relying on the data scientists and machine learning engineers to create a product that has clear input and output, is reliable to use, and won't surprise them in production. As a data scientist, it's your job to deliver that to an engineering stakeholder. You have to think like an engineer and try and understand what would be the best product that they need.

Engineers also rely on data scientists to do analyses to help power the tools they are building. Data scientists can look at data to help prioritize features, diagnose bugs in the engineering systems, and assess the performance of customer facing products like websites. In these situations, engineers are closer to business stakeholders, where they need the data scientists to get them the knowledge to make the right decisions.

Difficulties tend to arise with engineer stakeholders around the uncertainty of data science work. When developing a software product, it's generally the case that you set out to design an API or process, and you do it. You have clear tasks involved in the design and requirements around what it should do. In data science on the other hand, you have very little expectations going into creating a product. It's not clear what data you'll end up needing because you may not know what would be important for the model. It's not clear what the output will be because it can often depend on the model and its performance. And it's not even clear if the idea will be feasible because you could find that no model is accurate enough to meet business expectations.

Because of how unknown data science projects are at the start, engineering stakeholders are often taken aback by how little data scientists can promise early on. Thus, as a data scientist you'll need to take extra care to communicate the process so that engineers will be less surprised as things change. Make sure to communicate early and often about what the data science process is and how you're following it. By letting engineers in on the ambiguities of data science they'll be less likely to be surprised by them.

12.1.3 Corporate leadership

The executives in a company have similar backgrounds to business stakeholders (or engineers if they're leading a technology organization) but far larger spheres of influence. These directors, vice presidents, and chief corporate officers are guiding the organization, and they need data to do it. Data scientists are often tasked with sifting through data to provide them with the insight they need to do their jobs. Data scientists may also be accountable to them if they are involved in a large-scale project that machine learning is a critical component in.

Corporate leaders are extremely busy and have little time to understand details that don't affect them. This leads to immense difficulty in getting any time with them, and when that time is granted it is brief. When you meet with a high-level executive, they generally want to

get immediately to the point and understand the implications. This makes sense: they're extremely busy people and the less work they have to do at figuring out what someone is trying to say, the more they can focus on making the decisions.

Corporate leaders tend to work with data scientists when data is needed to make a major decision, or when the leader wants to have a better understanding of a part of the company. Sometimes work that is done for business stakeholders or other people is shown to higher and higher leveled people in the organization, until people at the top are seeing it. In these situations, the analysis and report may be polished again and again as it goes higher up. Other times, an executive may request a specific analysis or piece of work be done, to which it's the data scientist's job to create something that can be understood immediately by someone seeing the results for the first time.

Depending on the size and culture of the organization, a lot of work may have to be done before results are shared with an executive stakeholder. At some organizations, teams of people review results to ensure they align with business objectives and organizational beliefs. At more relaxed or smaller organizations, data scientists can create work directly for the leader. Regardless of the company, it's always the case that the work should be clean and without error.

Trouble tends to happen when work is presented that is unclear or incomplete. If the executive can't understand what's being presented to them they will have no patience to wait for things to become clear on their own. If they ask questions that can't be answered by the data scientist, they may feel that the work isn't reliable. If collaborating with an executive goes sufficiently badly and the executive is unforgiving, that could be a serious blow to the team.

On the other side of the coin, if the executive likes the results or finds them valuable, that can be a great boon to the data scientists. By becoming a trusted partner to an executive, a data science team can gain leverage in an organization to use data and machine learning in more places and for more reasons.

12.1.4 Your manager

Depending on the project at hand, your manager is sometimes a stakeholder. If your manager assigns you a task and continuously checks in on it and makes suggestions, they effectively are a stakeholder on the project. A manager wants a project to succeed because (1) their job is to make you succeed (2) it looks good for them if the projects assigned to their team do well and (3) the project may align with the broader goals your manager has for the team you're on.

In general, your manager should be guiding and mentoring you on your project. As you have struggles, you should be able to talk to your manager about them and your manager should help you find the best course of action. Your manager will take your work and help it go as far as it can in the business—by telling people about it, by helping the work get integrated into existing processes, and by thinking of new opportunities to improve your work.

But a manager is also a stakeholder because they are relying on you to do work. They need you to deliver the best work that you can because your reports, models, and analyses are what your manager shares. So, a manager fills a dual role: they both are a person you can rely on for help, but also a person who you have to do work for.

Because of this dual role, everything in the rest of this chapter relates to your manager too. The main difference is that with your manager you can let your guard down and show more vulnerability. It's reasonable to tell your manager something like "wow I'm really struggling to finish this analysis" but you probably couldn't reveal that to an executive stakeholder. A manager is able to treat you as more of a human being and provide advice, whereas other stakeholders are purely customers of your work.

Overall, treat managers just like you would the other people you work with—give clear updates, have constant communication, and deliver presentable work. But when you are having trouble, open up to your manager first, because they want to help you and understand when things need assistance.

12.2 Working with stakeholders

To effectively communicate with stakeholders during your data science projects, there are three core tenants to think about:

1. Understand the stakeholder's goals
2. Constantly communicate
3. Be consistent

Let's going into detail on each of these.

12.2.1 Understanding the stakeholder's goals

Everyone has goals when working a job—objectives they want to achieve when they go to work every day. Those goals are determined both by the job a person has and their personal traits like ambition and desire for career life balance. For example, a lead engineer may be focused on getting their current project complete so that they can get a promotion. Another possibility is a senior executive who knows they are going to leave the company soon and wants to avoid rocking the boat before then. These goals shape what we do when we are at work and how we respond to other people's actions. A project taking longer than expected could be terrible for the promotion-oriented engineer but great for the executive who doesn't want to do anything.

When working with stakeholders as a data scientist, it's critical that you understand their goals. The exact same analysis could be received greatly or poorly depending on the stakeholder's perspective. For example, consider an analysis on the performance of a certain product being sold on the company website. Suppose the stakeholder is the person who manages that product, and your analysis found that the product was not selling well in South America. If the stakeholder had the goal of making the product seem amazing in the company

since it was their idea, the analysis could be received very poorly since it shines a light on the problem. On the other hand, if the stakeholder had the goal of keeping a whole portfolio of products performing well, knowing which one might need to be cut is very helpful.

When working with a stakeholder, you should try and understand their goals and motivation as quickly as possible. The faster you can understand that, the less likely you are to deliver something that is poorly received unnecessarily. There are a few different ways you can discover a person's motivations:

1. **You can ask them directly.** By asking a stakeholder "what's important to you?" you're effectively opening a door for them to reveal themselves. What people openly say isn't the whole picture, but you can often pick up the essentials this way. Plus, it's a totally normal question to ask during an introductory meeting.
2. **You can ask around.** See if your coworkers have worked with the stakeholder before. Asking someone on your team a question like "so tell me about this stakeholder: what's their deal?" can get coworkers to fill you in. Keep in mind that you want to avoid gossiping—don't take what a coworker told you in confidence and spread it to others without thought.
3. **You can infer from their actions.** Sometimes it can become pretty clear from what the stakeholder is doing what their motivations are. For example, if you're presenting an analysis that shines poorly on one of the products they manage and they get overly defensive, that's a sign that product is extremely important to them. The downside of this method is that you have to learn through interacting so mistakes can easily be made, but if you're going to be interacting with them anyway you might as well spend time on learning from it.

By partaking in these tasks, you should be able to build a mental model of the stakeholder. How will they react to different outcomes of an analysis? Or delays in a model? If you can think through the outcomes in advance, you can be thoughtful in your communications.

Note that understanding the stakeholder's motivations doesn't mean you have to cater for them. While understanding their goals helps you predict how they will react, there may be plenty of cases where you chose to take actions that go against their goals because it better aligns with your motivations. If your goal is to be the best data scientist that you can, then in the example situation where your analysis shows a product does poorly it would be to your best interest to be honest in your analysis and not hide that finding. Knowledge of the stakeholder's needs is just something that can aid you.

If you find yourself in a situation where you have to deliver news that the stakeholder won't receive well, the first course of action is usually to call in reinforcements. Can your manager or a more senior member of your team help? By having someone to help you deliver this message, they'll be able to navigate any political fallout or issues rather than you. It isn't expected that a junior data scientist is able to be an expert in the politics of the company and the bigger picture.

In situations where you have to navigate a difficult conversation on your own, it's best to try and frame the conversation as a collaboration. Think about how you can be on the same side as the stakeholder. The news may be hard to hear, but you can try and convince them that you're not intentionally inflicting pain but trying to see the situation from their side and looking for areas of opportunity to move past it. In this case the conversation is much more of a traditional business negotiation and discussion than a technical one—it's about having to different sides come to a shared understanding.

Key Performance Indicators (KPIs)

Key Performance Indicators (KPIs) or OKRs, Objective Key Results, are metrics that a team or organization focuses on because they drive business value. For example, an online retail team may focus on orders per month as a number they want to raise. KPIs are a useful for data scientists since they provide an explicit quantification of the goals of the team. If you are able to find out a team's KPIs, you'll then be able to frame all of your analyses and other work in terms of "how does it affect their KPIs." If an analysis or method does not relate to a KPI, the team probably won't be interested. Not every team has core KPIs, and sometimes they are constantly changing or are poorly defined, but if you do have a situation where you are given KPIs its best not to ignore them. They're often the easiest way you can quickly understand your stakeholders goals.

12.2.2 Constantly communicate

It's easy for a data scientist to worry that they're either communicating too much or not communicating enough. "Is emailing a stakeholder for the third time in one day too much?" is a thought that may cross your mind as you hit send once again. Or just as easily you could worry: "I haven't talked to our stakeholder in a while, I wonder what they are thinking?" Perhaps worst, you could not put any consideration into how you're keeping the stakeholder in the loop.

For a data scientist, it is almost always the case that they are not communicating enough. Stakeholders thrive on communication. Emails, meetings, and calls are the only way that stakeholders can understand what is going on with a project. Without enough communication, a stakeholder can believe themselves to be out of the loop and worry that they don't have a good understanding of what is going on. And if there isn't enough communication, by the time a data scientist does talk to a stakeholder, the stakeholder can be taken aback by how different their expectations and reality are.

There are several different messages that a data scientist should be communicating to stakeholders. First, a data scientist should be keeping a stakeholder in the loop on how the project is meeting the expected timeline. If at the start of the project it seemed like it would take a month to find and clean the data and then a month to build a model, is that still the expected timeline? Ideally this should be a system where the data scientist shares changes and delays as they happen. A bad scenario is that the stakeholder is expecting a project to have finished already, but the data scientist has weeks or months of work ahead and hasn't

shared that information. When the stakeholder eventually finds out, they can be upset and rightly so.

A data scientist should also communicate how the project is progressing. That can be findings that the data scientist has discovered during the project, or areas where the data scientist is finding more difficulty than expected. Something like being stuck because of access to a database could potentially be resolved with stakeholder help. Share where the analysis is going well may help the stakeholder improve the project scope. If it seems like the project is faring very poorly, that should be communicated too (see the next chapter for more information on projects that fail).

Distinct from how the project is progressing, a data scientist should be constantly updating the stakeholder with how the work informs the business and what comes next. Data scientists should have opinions on how what has been done so far should change the project trajectory. If for example, a data scientist was doing an analysis and found something totally novel, then the data scientist should create a set of recommendations to the business on what to do with that finding.

Often the best way to create this consistent communication is to have it be the default way the project works. Nothing does that better than a recurring meeting on the calendar between the data scientist and the stakeholder. By having a weekly or bi-weekly meeting on the calendar, you're ensuring that there will be some required communication. This is a forcing function: by having it on the calendar you'll be forcing yourself to have to make something to share for each meeting. For each one of these meetings you should come prepared with a list of updates to the timeline, notes on what is going well or poorly, parts of your work to share, and suggested next steps.

As a data scientist you should also get in the habit of directly emailing stakeholders as needed. As an aspiring or junior data scientist it can be very intimidating to ask questions to senior people in the organization. However, most of the time stakeholders are happy to answer questions if it means your work will be better—that's their role in the organization. If you're worried that the person is so high up that the email needs to be extra polished, or if you think that the questions sound obvious enough that they may make you seem like you don't know much, run the email by your manager first—that's your manager's role. Depending on the stakeholder and the project you should be emailing them once a week or so.

If you find things are suddenly changing in your project and you need stakeholder input, like a dataset you thought existed doesn't, sometimes the right route is to set up a call or impromptu meeting with them. These sorts of quick gatherings can be great to get immediate input when necessary. The only question to ask yourself before getting one of these on the calendar is: "do I actually need stakeholder input on this?" If there is a change but you know what to do next, you probably don't need to disrupt other people's time. If you do require input, then go for it. A mistake that junior data scientists tend to make is having a default assumption that other people set up meetings, not them. But the more you can be proactive in keeping the project moving, the better the project will go. Further, this is great training for more senior roles where these actions are expected.

The method and reasons for communication should vary based on the stakeholder type. In general, business stakeholders tend to be happy with meetings where they can help provide direction and collaboration. They probably aren't the people who will be able to provide actual data or help with technical blockers. Engineers often have the technical answers for you but will be just as uncertain as you when making decisions about the project or direction the work should. Executives are extremely busy, and usually can only be in the loop at the beginning of the project to set broad goals, and the end of the project to see the conclusion.

12.2.3 Be consistent

Imagine there was a bakery down the street from you. One day you order a cake from them for a party and when you got the cake it was the best cake you ever had. A month later you order another cake, and this time it tastes decent but it's missing the frosting. You order a cake again and this time it's perfect, but they delivered it a week early. Does this sound like a bakery you'd want to keep going to?

Businesses thrive on delivering a consistent product, and as a data scientist you are a mini-business within your organization. The stakeholders are your customers and if you don't serve them well they will stop asking you for help. One way to deliver consistency in your relationships is through standardization of your work.

In the case of analyses and reports, you can do your stakeholders a great service by creating a consistent framework you share them in. If you can keep as much the same from analysis to analysis as possible, the stakeholders will be able to focus on the findings. Some things to consider standardizing:

- **Structure of the analysis**. As much as possible try and have a format around the analysis. Start with the same style of "objective and data" and end with similar "conclusions and next steps." You're training your stakeholders in how to read and think through these materials.
- **How the analysis is delivered**. While you don't have to exactly stick to it, things tend to go more smoothly if you have one file type for your analyses. They can be PowerPoint, PDF, html, or something else. They should all be stored in the same place each time. You can create a Dropbox, network folder, or other sharing tool for your analyses. Make sure it's something stakeholders can use—a GitHub repository probably won't work, although you could make one for yourself to keep everything under version-control.
- **Styling**. It seems small, but consistency in the visuals can go a long way. Use the same colors and templates as much as possible (bonus points if they're your company colors.)

When delivering dashboards, much of the same consistency rules as analyses apply. You want to keep the styling and format consistent between multiple dashboards and have all of them in a shared location so that people can remember how to access them.

For APIs and machine learning deliverables, the consistency is in the product's design. As a data science team's portfolio of APIs and models grows, it can be extremely hard to keep track of how each one works. The more the APIs are consistent with each other, the easier they are to use. This includes:

- **Consistency in the input**. How your models and APIs take data in should follow the same format as each other as much as possible. For instance, they can all take JSON objects with the same names for parameters in them.
- **Consistency in the output**. How the output is structured should both make sense with how the input is structured and how the rest of the APIs created by the team work. If the model takes JSON as input, it should return JSON as output.
- **Consistency in authentication**. It is likely that the models and APIs require some form of authentication for security. Whatever method is used it should be consistent across as many of the APIs as possible, especially because it's easy to lose track of which credentials are for which API.

In addition to helping out your stakeholders, all of this consistency will be useful to you! The more you can standardize all of these parts of the data science work, the less you'll have to think about them (and the more you can focus on the interesting parts). The more a data science team can standardize, the easier it is to pass work between different people. Standardization is good for everyone.

12.3 Prioritizing work

As a data scientist trying to support an organization, you often have to decide what task you should be working on. While some teams have a project manager who decides what work each data scientist should do, you should still be recommending what the next task is. These tasks can vary wildly in topic and scope and may each come from a different stakeholder. You can classify tasks into one of three buckets:

- **Quick tasks directly from stakeholders**. These tend to be small requests like "make a graph of sales over time." They are often urgent and since they don't take much time it's hard to say no. But each one is a distraction from more important work and as they build up it's harder and harder to do productive work.
- **Long term projects for the business**. These projects are the core of a data scientist's job. Building dashboards, making long-form analyses, and creating models to put in production all fall into this bucket. These tend to be highly important but given that they can take weeks or months they are not always urgent.
- **Ideas that you personally think have a long-term benefit**. Given the nature of being a data scientist these generally are more technical, like creating a machine learning model to predict when a customer will call support before they do. This also includes work that makes you more productive, like creating functions or even a library to more quickly solve common problems you encounter. If there is a manual process

that takes hours of time each week to run, you can automate the task which provides no direct benefit to the business but indirectly frees you to do more. No one is asking for this work, but it feels important.

It can be difficult to figure out what's the most important task be working on and what can be put on the back-burner, especially when you have multiple people requesting work from you. At the same time, the work that stakeholders think is very important to them may not be important for the business as a whole. As a data scientist you rarely have the ability to decline requests from stakeholders, since usually they are the one driving the direction of the business. All this creates an environment where the decision of what you work on can hugely influence the business, but also you are constrained in what you can choose to do.

This is an area many data scientists struggle with. When stakeholders make requests, it's natural to want to please them and not to disappoint. Plus, the requests they make can be intellectually interesting. However, trying to fulfill every request is unsustainable since the requests for answers are endless. Further, answering one question with data often leads to new questions, so fulfilling requests often creates additional work rather than lowering the amount of work left to do.

In this situation when considering possible tasks to work on it helps to focus on two questions.

- **Will this work have an impact?** Does knowing the result of this analysis materially affect the company? Is there any result that would change decisions? Will this machine learning model increase profit?
- **Will this work do something new?** Is this applying an existing process over and over, or is it trying something different?

These two questions create four different combinations of types of work:

1. Innovative and impactful
2. Not innovative but impactful
3. Innovative but not impactful
4. Neither innovative nor impactful

In the next few sections we'll go into each one of these in more detail.

12.3.1 Both innovative and impactful work

Innovative work that alters the business is what most data scientists want to spend their whole career doing. An example project would be something like taking inventory data that has never been touched by a data scientist before and using a start-of-the-art machine learning model to optimize ordering product and saving the company millions of dollars. They're the kind of projects that at their best get you featured in magazines like the Harvard Business Review or Wired.

Unfortunately, not many projects fall into this category at companies. For these projects to exist there needs to be many different things going for it. There needs to be enough data that

data science methods can be useful. There has to be an interesting signal in the data that the models can pick them up. And the part of the business has to be large or important enough that changes can make a difference to the bottom line (so probably not optimizing dry-erase marker inventory for the office). And lastly, the problem must be complex or unique enough that people haven't tried it before. The set of problems at a company that fall into all of those buckets is exceedingly small.

These projects are great because they have excitement from both the stakeholders and the data scientists. The stakeholders are feeling great because they can see the clear value from the project. The data scientists are eager to try out using new methods on new data and seeing the results. If you find a project in this category, do everything you can to nurture it. These are the sorts of projects that can define a career, but because they have so many requirements they are very rare and don't often succeed.

12.3.2 Not innovative but still impactful work

These are the projects that aren't innovative but alter the business—like doing a very mundane data analysis that convinces a team to launch a product. Often this accounts to providing proof of a thing that everyone suspects is true—it's not particularly innovative but it will help out. Or in the case of engineering projects, these could be projects like taking a model that has already been deployed in one division of the company and redeploying it to another division. Another type of work that falls in this category is streamlining tasks that take lots of time. That kind of work isn't innovative, but it does improve the business.

While this work isn't glamorous, what's important is that the work is helping the business. Having stakeholders be able to see the value of data science work is great for getting buy-in. If you have more buy-in, the next time a project goes over budget or doesn't pan out people are more likely to keep their faith in you. So as much as possible, try and take on these projects.

At the end of the day the job of a data scientist is to provide value to the company, not to do the work that's the most fascinating to them. A valuable skill for data scientists is to be able to tolerate this kind of valuable-but-not-interesting work. That being said, if a job is entirely full of projects that are not interesting you and where you aren't learning anything, it's entirely appropriate to seek a different job. It's totally valid to consider job satisfaction when prioritizing work, but just make sure it's not the only consideration.

12.3.3 Innovative but not impactful work

This is work that is innovative but not useful to the business—such as researching new theoretical data science algorithms and methods that have a low chance of being used. These projects can be "ivory towers," where people spend months or years holed up on work that doesn't interact with other groups and won't end up being used. These projects can be huge time sinks for data scientist teams and end up costing millions of dollars with little to gain. Despite all this, these projects attract data scientists to them like moths to a bright light.

These projects tend to start within the data science team and be focused on what is methodologically interesting rather than what is useful for the business. For example, a data scientist could read a research paper outlining a new theoretical technique and convince the rest of the team that they *must* try using it on their own data. Six months later, it's clear the method isn't as good as the paper suggested, and even if it was no one in the business has a particular need for the results that algorithm would have provided. Worse, the data scientist has since moved on to a new paper and the process repeats.

Often stakeholders don't even know about these projects. At most they notice that some of the data scientists on the team seem very busy working on something that sounds really hard but no one has explained what it is. As a data scientist, it's easy to feel that once you complete the project then people will be able to find a use for it. In practice, if you can't immediately see a use for the project then stakeholders probably won't be able to either. As much as possible don't get stuck working on these projects, since they will lead you to look like they aren't contributing to the business, and people will question your value.

12.3.4 Neither innovative nor impactful work

Unfortunately, plenty of work that data scientists get ask to do is neither innovative nor impactful. The most classic example of this is a frequently-updated report that isn't automated and takes a long time to make, and yet no one bothers to read every time it's published. This sort of work takes a ton of time and effort to make, and yet if it's delivered to many stakeholders no one is willing to be the stakeholder who steps up and says it doesn't need to be done. As the business collects more and more required reports over time, the time it takes to generate all of them can eventually weigh down a data scientist team.

While reporting is one type of work that has the potential to be neither innovative nor impactful, plenty of small one-off requests can fall into this category too. Executives who like data and charts can repeatedly make requests to the data science team like: "make me a chart of sales in Europe by week" or "find me the product that has the biggest drop in orders in the last twelve weeks." None of these requests may be particularly difficult, but together they take a lot of bandwidth, and probably don't provide very much value to the business.

These sorts of situations are difficult because there are no easy answers. For reports and processes that take lots of time you can try automating them, but that itself takes a lot of time too and you may only get limited improvements depending on the technology being used. For high-level stakeholders making repeated requests, it's hard to say no without it jeopardizing the data science team's standing.

Despite the situations being difficult, it's your responsibility as a data scientist to try and advocate for your time being well used. If there are many of these tasks going on, you should be making it clear to other people that it might not be worth the time. This could be conversations with your manager or your stakeholders. It's likely that they already know this work isn't especially useful, however by having continuing discussions around the processes and how you think they should be improved then people will be less willing to accept the

status quo. If not, then sometimes the best thing you can do is try and make your suggested improvements first and then show them off.

12.4 Concluding remarks

Working with stakeholders is a constant process throughout the course of a project. You need to understand their needs and why they are asking them. Not only does a project start due to stakeholders having a request, but what they are requesting will likely change during the project itself and its your responsibility to keep up with that. The more you can align your project with what they are asking for the less likely it will fail. In the next chapter we'll talk about what happens when a data science project fails, and one way that can happen is when the stakeholder communication breaks down.

Turning requests into dialogues - Sam Barrows, Data Scientist at Airbnb

One valuable tool for working with stakeholders is to turn requests into dialogues. You may often have colleagues ask you to complete specific tasks. Rather than immediately accept or reject these requests, start a dialogue about why the request is being made. What business need does the request aim to solve? Is there a better way to achieve the intended outcome? By understanding the motivations behind the requests that you receive, you are more likely to do meaningful work.

This strategy is practicing interest-based negotiation, where the players in a negotiation focus on addressing their underlying interests, rather than just their more immediate needs. In this case, the immediate needs are the requests that you receive, while the underlying interests are the business motivations behind these requests, as well as the objectives of the data science team.

12.5 Interview with Sade Snowden-Akintunde, a Data Scientist at Etsy

Sade Snowden-Akintunde works at Etsy, where she specializes in experiment design and analysis to improve purchasing experiences for international consumers. Her areas of expertise include A/B testing and experimentation, implementing reliable data practices, and scaling data infrastructure.

12.5.1 Why is managing stakeholders important?

Unfortunately, it doesn't matter how smart you are if you cannot communicate concepts to non-technical stakeholders. At the end of the day, a lot of these companies are run by people who may not have the same level of technical skill as you. You need to be able to communicate to them in a way that makes them feel both capable and allows you to advocate for yourself if necessary. Managing stakeholders is probably one of the most important aspects of data science, but often it's highlighted the least.

12.5.2 How did you learn to manage stakeholders?

Trial and error: I've had situations that have worked and situations that did not work and I paid attention. I think the biggest thing that I've learned is it's really important to communicate early and repeat yourself to make sure that people understand what you're saying. Early in my data science career, I assumed that if I said something once and somebody agreed then they knew exactly what I was talking about, but people may not even know that they don't understand what you're saying.

12.5.3 Was there a time where you had difficulty with a stakeholder?

Early in my career, I was afraid to advocate for myself and from my perspective as an experiment designer. Other people would make experiments that did not make sense to me, but I didn't say anything. Then I would try to analyze the experiments after they finished, and the results would be difficult to interpret because of the experiment design. I should have been working with the stakeholders from the very beginning to communicate how I could best analyze their work and what they would get from their experiment with proper design. I realized I have to actually say something from the very beginning if I want to be able to do my best work at the end.

12.5.4 What do junior data scientists frequently get wrong?

I think junior data scientists assume people are going to automatically recognize the value in your work. This is especially common among data scientists who come from academic backgrounds. We tend to get really wrapped up in everything being super thorough and following the scientific method. While this is important in academia, just working hard is not necessarily what's going to get people to recognize the value of your work. The way that you communicate is what gets people to recognize the value of your work.

12.5.5 Do you always try to explain the technical part of the data science?

It depends on how much the stakeholder wants to be involved. I work with project managers who did not want to be involved in anything technical. If I just said "this is not working right now" they would take that at face value. I've also worked with project managers who wanted to know every detail and what I have found is that they do tend to get a little overwhelmed. Some people want you to check in on a regular basis and tell them what's happening and even if they know that they don't understand they just want to feel in the loop. So I'll just make sure that they feel in the loop.

12.5.6 What's your final piece of advice for junior or aspiring data scientists?

I think that people tend to want to go into technical careers like data science because they think that you can focus only on the logical element and not deal with the human element. But that's not the case at all. When people are considering a career in data science, they should

really think about if they are willing to have small ego in order to communicate and do their job well. It's really easy to say, "I want to learn how to build this model and I want to learn A/B testing and I want to learn all these technical things." While that’s great, the soft skills are what are going to take you far in your career.

12.6 Summary

- Stakeholders come in many forms, with many different needs
- Create relationships with stakeholders where they can consistently rely on you
- Have constant communication and keep stakeholders in the loop on updates and difficulties with projects

Part 4: Growing in your data science role

The final part of this book is material for after you get comfortable in a data science position—it's all about what comes next. The topics in this part are ones that eventually affect every data scientist yet aren't frequently talked about. It's easy to assume that if you have a stable data science job you've "made it," but there is a lot more to learn. The goal of this final part is to provide you with materials to move your career from a junior data scientist to a senior data scientist and beyond.

Chapter 13 covers how to handle when data science projects fail. This topic is critically important for more veteran data scientists, because as your career progresses you'll certainly run into failures. Chapter 14 is about joining the data science community—from writing blog posts to attending conferences. While not required to be a data scientist, community involvement can be hugely beneficial for building a network and landing future jobs. Chapter 15 handles the tricky task of leaving a data science position in a way that is best for your career. As the last chapter of this book, Chapter 16 discusses some of main career paths after senior data scientist, such as becoming a manager or a technical lead.

13

When your data science project fails

In this chapter we cover

- Why data science projects tend to fail
- What you can do when your project fails
- Handling the negative emotions from failure

Most data science projects are a high-risk ventures. Either you're trying to predict something no one has predicted before, optimize something no one has optimized before, or understand data that no one has looked at before. No matter what you're doing, you're the first person doing it—it's almost always exploratory. Because data scientists are continuously doing new things, you will inevitably hit a point where you find out what you hoped for just isn't possible. We all must grapple with our ideas not succeeding. It is heartbreaking, gut-wrenching, and you just want to stop thinking about data science and daydream about leaving the field altogether.

As an example, consider a company building a machine learning model to recommend products on the website. The likely course of events starts with some set of meetings where the data science team convinces executives it is a good idea. The team believes that by using information about customers and their transactions they can predict which customers want to buy next. The executives buy into the idea and green-light the project. Many other companies have these models and they seem straight forward so it should work.

Unfortunately, once the team starts working on it, reality sets in. Maybe they find out that since the company recently switched systems, transaction data is only available for the past few months. Or maybe you run an A/B test and find the people who see the recommendation

engine don't buy anything more than the people who don't. Problems like these build up and eventually the team abandons the project, dismayed.

For this chapter, we'll define a project as a failure when it doesn't meet its objective. In the case of an analysis that might mean the project doesn't help the stakeholder answer the business question. For a production machine learning problem, a project could fail when it isn't deployed or when deployed doesn't work. There are many different ways projects can fail.

Data scientists tend to not talk projects failing, although it happens extremely often. When a project fails it can make a data scientist feel vulnerable. If your project fails, you may think to yourself, "had I been a better data scientist this wouldn't have happened." Few people are comfortable sharing with others stories about when they questioned their own abilities.

At its core, data science is research and development. Every day data scientists take data that has never been analyzed before and search for a trend that may or may not be there. Data scientists set out to build machine learning models on data where there may not be a signal. It's impossible for these tasks to always succeed because new trends and signals are very rarely found in any field. This is different than a field like software engineering where given a task it is usually possible to complete it (although perhaps taking more time and resources than you planned).

Understanding how data science projects fail and what to do when they fail is important. By better understanding a failed project, future failures can be avoided. Failed projects can also give insights into what will succeed, by investigating what parts of the project did work. And a failed project might be able to be adjusted with a little work into something that could be useful within an organization.

In this chapter we'll cover three topics: why data science projects fail, how to think about project risk, and what to do when a project is failing. We will discuss three main reasons why most projects fail, what to do with the project, and how to handle the emotions you might feel.

13.1 Why data science projects fail

It can seem like data science projects fail for an endless list of reasons. From budget to technology failing and tasks taking far longer than expected, failure can come from many reasons. Ultimately these many types of failure break down into a few core themes.

13.1.1 The data isn't what you wanted

You can't look into every possible data source before starting a project. It's imperative to make informed assumptions on what is available based on what you know of the company. Once the project starts, you often find out that many of your assumptions don't hold. Either data doesn't actually exist, it isn't stored in a useful format, or it's not stored in a place you can access. For example, if you're doing an analysis to understand how a customer's age affect their use of a loyalty program, you may find out that customers are never asked their age when joining the program. That can end a project very quickly.

Example failure: analysis on loyalty program status

A director in the marketing department of a large restaurant chain wants to understand if customers spend differently as they increase in status of their loyalty program. The program has silver, gold, and platinum levels, and she wants to know if someone who hits platinum bought in the same way when they were merely at silver.

The data science team agrees to look into this, since it should be a fairly straightforward task and they haven't worked with the loyalty data before. They're shocked to find the antiquated loyalty program database they use doesn't keep track of historic loyalty program levels, just what a customer is now. If a customer is currently at the platinum level, there would be no way to know when they used to be silver or gold. Thus, the analysis is impossible to do.

The data science team recommends the system be adjusted, but changing a loyalty program database architecture requires millions of dollars and there is little demand for it in the company, so no changes are made and the analysis idea is abandoned.

Since you need data before you can do anything, these are the first major problems that arise. A common reaction to running into this issue is an internal bargaining where you try to engineer around the holes in your data. You say things like: "well we don't have a decade of data like we wanted, but maybe a year of data will be sufficient for the model" and hope for the best. Sometimes this can work, but sometimes the alternate solutions aren't adequate enough for the project to be feasible.

When you pitch a project you don't always have access to the data, or even a full understanding of what it is (this is especially a problem in consulting where you don't get access to the data until the work for the project is sold). Further, the data may exist but have a critical flaw that renders it useless. The data might exist in a database table, but the customer ids might be corrupted and unusable. There are so many ways a dataset could have problems that it's extremely difficult to check them all before starting a project. Because of this, it's common for data science projects to barely get past the launch phase.

The quicker you can get access to the data and explore it, the more quickly you can mitigate the risk of inadequate data. The best-case scenario is to get samples of data before starting a project to avoid this error. If that isn't feasible, the next best scenario is that the project timeline is designed around the possibility that the data is poor. By having an early "go/no go" step in the project where the stakeholders agree to reassess the project feasibility, there's less of a chance of stakeholders being surprised that the data could be bad.

If you find yourself with a project struggling with a lack of good data, you have limited options. You can try and find alternative data sources that could possibly substitute—for example, maybe you don't have data on what products were purchased, but you do know what product volume was manufactured and you can use that instead. The problem is usually these substitutes are different enough to cause real problems with the analysis.

In cases where there isn't a viable substitute, sometimes all you can do is start a separate project to begin collecting better data. Adding instrumentation and telemetry to websites and apps, creating databases to store data instead of just throwing it out, and other tasks can help the team in the future take on the task with better data collected.

13.1.2 The data doesn't have a signal

Suppose a gambler hires a data scientist hoping to use statistics to win a dice game. The gambler rolls a six-sided die 10,000 times and records the rolls, then pays the data scientist to create a model to predict the next die roll. Despite the data scientist having an immense amount of data, there is no way the data scientist could predict what roll will come next beyond assigning each side a $1/6^{th}$ probability (if the die is fair). Despite the data scientist having lots of data, there is no signal within that data as to what side would be rolled next.

This problem of not having a signal in the data is extremely common in data science. Suppose you were running an ecommerce website and wanted to create a model to predict which customers would order based on their browser, device, and operating system. There is no way to know before starting a project if those datapoints could actually be used to predict if a customer would order, or if it lacks a signal just like the die roll data did. The act of creating a machine learning model to make a prediction is testing the data to see if there is a signal within it, and there very well may not be. In fact, in many situations it would be more surprising for there to be a signal than there would be for there to not be a signal.

Example failure: detecting bugs on a website with sales data

A hypothetical ecommerce company has a problem: the website keeps having errors and bugs. Worse, the errors aren't always detected by DevOps or the software engineering team. Once, the error was detected by the marketing team noticing daily revenue is too low. When marketing is the one realizing there is a bug then it's a bad situation.

The data science team sets out to use statistical quality control techniques on the sales data, so that they can have alerts when revenue was so low that there must be a bug on the site. They had a list of days when bugs were detected and historic revenue data. It seemed straightforward to use sales to predict bugs.

Unfortunately, the amount of reasons revenue can change on a daily basis made detecting bugs from it almost impossible. Revenue could be low because of the day of the week, the point in the year, promotions from marketing, global events, or any other number of things. While once marketing was able to see a bug impact once, that wasn't generalizable because there wasn't a signal for it in the data.

Unfortunately, not having a signal in the data can be the end of the project. If a project is built around trying to find a relationship in the data and make a prediction on it, if there is no relationship there then the prediction cannot be made. That can mean an analysis turns up nothing new or interesting, or a machine learning model fails to have any results that are better than random chance.

If you find yourself in a project where you can't seem to find the signal in the noise there are a couple of possible ways out. First, you can try and reframe the problem to see if a different signal exists. For example, suppose you had a set of articles, and you were trying to predict the most relevant article to the user. You could frame it as a classification problem to try and classify which article in a set of articles is the most relevant. Alternatively, you could frame it as a prediction problem where given a single article, what is the probability it is

related to another one. These framings end up with quite different modeling approaches and the way you're thinking about the task can make a meaningful difference.

If nothing is seeming to pull a signal out of the data, you can try changing the data source. Just like with the previous failure point of not having good data, sometimes adding a new data source to the problem can create an unexpected signal. Unfortunately, usually it's the case you start with the dataset that had the highest chance of being useful, so the odds this strategy will save you are fairly limited.

It's usual for data scientists stuck in this situation to try and use a more powerful model to find a signal. If a logistic regression can't make a meaningful prediction, then try a random forest model. If a random forest model doesn't work then try a neural network. Each method ends up being more time consuming and more complex. While these methods can be useful for getting more accurate predictions, they can't make something out of nothing.

Most often it's the case that if the simplest method cannot detect any signal, the more complex ones won't be able to either. Thus, it's best to start with simple modeling methods to validate the feasibility of the project and then move to more complex and time-consuming ones, rather than start with the complex ones and go simpler. Don't get lost spending months building more and more complicated models hoping that just maybe the next one will be the one that saves the project.

13.1.3 The customer didn't end up wanting it

No matter how accurate a model or analysis is, what matters is that it provides value to the stakeholder. An analysis can have findings that are incredibly interesting to the data scientist but not to the business person who requested it. A machine learning model can make highly accurate predictions, but if that model isn't deployed and used it won't provide much value. Many data science projects fail even after the data science work has been done.

Ultimately a data science analysis, model, or dashboard is a product. Designing and creating a product is a practice that many people have put hundreds of years of collective thought into. Despite all of that, every year billions of dollars are spent creating products that people don't end up wanting. From New Coke to Google Glass, high profile products don't land with customers, and low profile ones are the same way. And so just like how Microsoft and Nokia can put lots of effort into creating Windows Phone that customers don't end up buying, so too can a data scientist create products that aren't used either.

Example failure: sales and marketing campaign value prediction

A project at a retailer is started to create a machine learning model that predicts how much return on investment (ROI) future advertising campaigns will bring. The data science team decided to build the model after seeing how much the marketing and sales teams struggled with making Excel spreadsheets that predicted the overall value. Suppose that by using machine learning and modeling at the customer level, the data science team created a Python-based model that more accurately predicts the return on investment of the campaigns.

Later, the data science team finds out that the only reason why the marketing and sales teams created Excel spreadsheets with ROI predictions was to get the Finance department to sign off on them. The Finance team refuses to

work with anything but Excel—Python is too much of a black box for them. Thus, the Python tool that the data science team created was never used and the work was fruitless. This was all because the data science team didn't consider the customer's needs—it wasn't a highly accurate prediction, it was a prediction to convince finance.

The universal guidance on avoiding customers not liking your product is to spend lots of time talking to and working with your customers. The more you can understand their needs, their desires, and their problems, the more likely you are to make a product that actually solves it. The fields of market research and user experience research are both different ways of solving this—either through surveys and focus groups in market research or through user stories, personas, and testing in user experience research. Many other fields have come up with methods too and have been doing this for years.

Despite all this good thinking people have done, data science as a field is especially susceptible to this kind of failure. For whatever reason, data scientists are much more comfortable looking at statistical test results, tables, data, and plots then they are going out and talking to people. Many data science projects have failed because the data scientists didn't put enough effort into talking to the customer and stakeholder to understand what their true problem was; instead the data scientists jumped to building interesting models and exploring data. In fact, this is one of the main reasons why we chose to write Chapter 12 on managing stakeholders. Hopefully you already have a better understanding of how to think through stakeholder relationships from reading it, but if you skipped it maybe check it out.

If you're finding yourself in this situation, the single best thing you can do is talk to your customers. It's never too late to talk to your customers. Whether your customer is a business stakeholder or literally customers of your company, the communication and understanding can be helpful. If your product isn't useful to them, can they tell you why it isn't? Are those problems you could potentially fix by adding new features to the product? For example, maybe an analysis could be changed by joining a different dataset to it. Maybe a machine learning model could be improved by adjusting the format of the output or how quickly it runs. You'll never know until you talk to people.

This also feeds back into the concept of a Minimally Viable Product (MVP) which is used heavily in software development. The idea because the more quickly you can get a product working and to market, the more quickly you can get feedback on what works or doesn't and iterate on that feedback. In data science, the quicker you can have any model working or any analysis done, the quicker you can show it to customers or stakeholders and get feedback. Spending months iterating on a model delays this from happening.

The more you can understand the customers throughout the design and building process of your work, the less likely you are to get this kind of failure. And if you have ended up failing in this way, the best way forward is to start communicating to try and find a solution.

13.2 Managing risk

Some projects are riskier than others. Taking data the team has worked with before and making a standard dashboard in a standard way is pretty likely to succeed. Finding a new data set in the company, building a machine learning model around it that will run in real time, and displaying it in a pleasant user interface to the customer is a risky project. As a data scientist, you have some control over the amount of risk you have at any time.

One big consideration with risk is how many projects you are working on at once. If you are working on a single risky project and that project fails, it can be quite difficult to handle that career wise. In a situation where the main project you're working on fails, it can be harder to find successes to share with your stakeholders and yourself. If however you are able to work on several projects at once, you will be able to mitigate the risk. In the situation where one of those projects fails you will have other projects to fall back on. If one project you have is an extremely complex machine learning model that has a limited chance of success, you could simultaneously be working on simpler dashboarding and reporting—if the machine learning project fails your stakeholders may still be happy with the reports.

Having multiple projects can also be beneficial from a utilization standpoint. Data science projects have many starts and stops—from waiting for data to waiting for stakeholders to respond and even waiting for models to fit. If you find yourself stuck on one project for some reason you'll have an opening to make progress on another. This can even help with mental blocks—distracting yourself when you're stuck can be a great way to refresh your thinking.

Another way to mitigate risk is to bake in early stopping points into a project. Ideally, a project that seems like it may fail should be designed with the expectation that if by a certain point it isn't successful it's cut off. For instance, in a project where it's unclear if the data exists, the project can be scoped so that if after a month of searching good data can't be found, it's considered infeasible and scuttled. If the expectation it might not work out is presented early then ending the project is less surprising and less costly.

In a sense, having the project end early is codifying the fact that data science is research and development. Because data science is fill with so many unknowns, it makes sense to plan in the possibility that as more is learned through exploratory work the idea may not pan out.

While it's worthwhile to minimize the risk in a project portfolio, you don't want to remove it entirely. Data science is all about taking risks: almost any sufficiently interesting project is going to have plenty of uncertainty and unknowns. Those risky unknowns can be because no one has used a new dataset before, or no one in a company has tried a certain methodology before, or the stakeholder is from part of the company that has never used data science before. Plenty of valuable data science contributions at companies have come from people trying something new, and if as a data scientist you try and avoid projects that could fail, you're also avoiding the potentially big successes.

While this chapter has covered many ways data science projects have failed, data science teams can eventually fail in aggregate by not taking enough risks. Consider a data science team that comes up with some new project ideas and reports, finds them successful, then

stagnates by only updating and refreshing the previous work. While those projects might not fail in that they are delivering work to the company, that team would miss the potential new areas for data science.

13.3 What you can do when your projects failed

If your data science project has failed, that doesn't mean all the time working on it was wasted. We outlined some potential actions you can take in the previous section to turn the project around. But even if there's no way it can succeed, there are still steps you can take to get the most out what's left of the project. We'll also give you some strategies for handling your own emotions about having a project fail.

13.3.1 What to do with the project

While the project may have failed, there likely is still a lot that can be gained from the project, both in knowledge and technology. These steps can retain much of that.

DOCUMENT LESSONS LEARNED

The first thing to do with a project that has failed is assess what you can learn from it. Some important questions to ask yourself and the team are:

- Why did it fail? This question seems almost obvious and yet it's often the case that you can't understand why a project failed until you step back and look at the bigger picture. By having a discussion with all the people involved in the project, you can better diagnose what went wrong. The company Etsy popularized the concept of a blameless postmortem in software development—a discussion after something failed where you can diagnose the problem without blaming a person. By thinking of a problem as caused by a flaw in the system of how the team works (instead of a person) you can more clearly find a solution. Without the fear of punishment in the discussion people will be more willing to talk about what happened openly.
- What could have been done to avoid the failure? Once you understand the factors that attributed to the failure, you can understand how similar situations could be avoided in the future. For example, if the data wasn't good enough for the project to work, then it could have been avoided by a longer exploratory phase. These sorts of lessons will help your team grow and mature.
- What did we learn about our data and our problem? Even if the project is a failure, you often learn things that can be valuable in the future. Maybe the data didn't have a signal in it but to get to that point you still had to join a bunch of new data sets together—now you can do those same joins more easily on other projects. These questions can help you brainstorm possible things that can be salvaged from the project and help brainstorm alternate project ideas.

By having a meeting where the team works through these questions and then saving the results in a shared location you'll get a lot more value out of the failed project.

CONSIDER PIVOTING THE PROJECT

While the project itself might have been a failure, there may ways to pivot it into something useful. For example, if you're trying to build a tool to detect anomalies in company revenue and it fails, you may still be able to use that same model as a pretty decent forecasting tool. Whole companies have been built on taking one idea that was a failure and repurposing it into something more successful.

Pivoting a product requires a lot of communication with stakeholders and customers. You're basically back at the beginning of the product design process trying to figure out a good use for your work. By talking to stakeholders and customers, you can understand their problems and see if your work is useful for anything new.

END THE PROJECT (CUT AND RUN)

If you can't pivot the project, the best thing you can do is end it. By definitively cancelling the project you allow yourself and the team to move onto new, more promising work. It's extremely easy as a data scientist to want to keep working on a project forever with the hopes that at some day it'll work (there are thousands of algorithms out there, eventually one will work, right?). But by getting stuck trying to get something to work you end up spending unnecessary effort. It's also not fun to work on the same thing until the end of time! While cutting a project is hard as it requires you to admit that it's no longer worth the effort, it pays off in the long run.

COMMUNICATE WITH YOUR STAKEHOLDERS

A data scientist should be communicating with their stakeholder throughout the course of a data science project (see Chapter 12), but they should increase the amount of communication if the project is failing. While it might feel comfortable to hide risks and troubles from stakeholders to avoid disappointing them, running into the situation where a stakeholder is surprised to find the project has failed can be catastrophic for a career. By letting the stakeholders know that problems are occurring or that the project is at the point where it can no longer move forward, you're being transparent with the stakeholder and inspiring trust. After helping them understand the project state you can work together to decide the next steps.

If you're uncertain how to communicate the problems with your stakeholder, your manager should be a good resource to help with this. They can either brainstorm an approach to delivering the message, or potentially take the lead on delivering it themselves. Different people and organizations like messages delivered in different ways—from spreadsheets laying out the issues with green/yellow/red color coding to conversations over coffee. Your manager or other people on your team should have insight to what works best.

It's common for a data scientist to have anxiety when communicating that a project is failing—it feels very emotionally vulnerable and like you are in a position of weakness. While there are occasions where the news is received poorly, it's often the case that other people will be willing to help work with you to resolve issues and decide next steps. After communicating the project failing you may feel relief, not suffering.

13.3.2 Handling negative emotions

Forget the project and the company for a bit: you also need to think about your own well-being. Having a project fail is emotionally difficult! It's the worst! If you're not careful, a failed project can be a real drain on yourself and haunt you long after the project is over. By being thoughtful with how you react to the failure and the story you craft about it, you can set yourself up for more long-term success.

A natural internal monologue at the end of a failed project is to think "if I were only a better data scientist the project wouldn't have failed." This is a fallacy—most data science projects fail because data science is inherently based on trying things that could never work. Most great data scientists have been involved with, or even led, projects that haven't succeeded. By placing the blame for the failed project on yourself and possible data science deficiencies you're putting the weight of the whole project on yourself. But as listed earlier in this chapter, there are many reasons that data science projects fail, and it's very rare that the issue is "the competency of the data scientist." Things like inadequate data, poor leadership, or no signal and only noise are things that are totally outside the control of any data scientist. It's very common to have an anxiety that the project is failing because if yourself. But consider that is an anxiety in your head and not a reflection of reality.

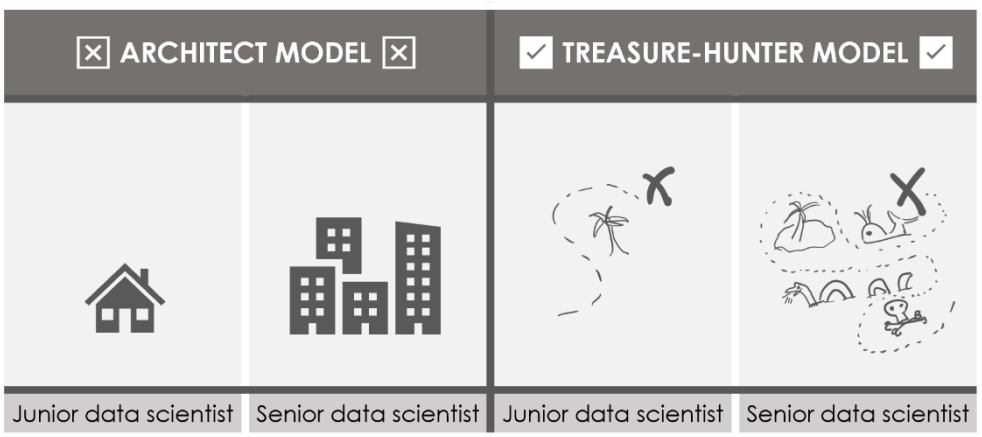
If you allow yourself to fail and accept that it isn't a sign of a weakness, you will be more able to learn from the experience. Being confident in yourself and your skills makes it easier to think about the failure and what contributed to it because it won't hurt as much. That being said, the ability to be confident and own a failure is one that takes time, patience, and practice to gain, so don't be surprised if you're struggling to have confidence. It's okay!

The key point here is that the best thing you can do for yourself when a project fails is understand that failure is not a reflection on your skills: projects fail for reasons outside of your control, and you'll be able to move on from it. The more you're able to hold those things close, the easier the failure will be.

Let's end this chapter with a metaphor for data science. It's common for aspiring and junior data science to think of a professional data scientist as like an architect of buildings. A novice architect may design simple homes, while an experienced architect can build skyscrapers—but if any of them has a building collapse it's a career ending failure. Similarly, one way to view a data scientist is that they build more and more complex models, but if one fails their career is jeopardized. After reading this chapter you hopefully recognize that *this isn't an accurate model of what a professional data scientist is*.

A better metaphor is a data scientist is like a treasure hunter. A treasure hunter sets out looking for lost valuables, and if they're lucky they'll once and a while find some! A novice

treasure hunter may look for standard goods while an experienced hunter finds the most legendary of treasure. A data scientist is much more like a treasure hunter—they seek out successful models and once in a while their models and analyses work! And while a senior data scientist might work on more complicated or tricky projects, everyone is continuously failing and that's just part of the job.



Two metaphors for data science, architecture vs treasure-hunting

13.4 Interview with Michelle Keim, Head of Data Science & Machine Learning at Pluralsight

Michelle Keim leads the Data Science & Machine learning team at Pluralsight, an enterprise technology learning platform with a mission to democratize technology skills. Having previously grown and lead a data science team at a range of companies including Boeing, T-Mobile, and Bridgepoint Education, she has a deep understanding of why data science projects can fail and how to handle it.

13.4.1 What was a time you experienced a failure in your career?

I got pulled in to lead a project to build a set of customer retention models. I thought I had talked with all the right stakeholders and understood the business need, how the team worked, and why the models were necessary. We built the models, but soon learned that there was no interest in them. The problem was we hadn't sat down with the customer care agents who would actually be using the output--I had only ever talked to the leaders. We delivered a list of probabilities of whether a customer would leave, but the care agents didn't know what to do with that. They needed to know what they should do when a customer is at risk of leaving, which is a very different problem than the one we had tackled. The biggest

lesson learned for me was that you really have to get down into the weeds and understand the use case of the problem. What's the problem being solved by the people who will use the output?

13.4.2 Are there red flags you can see before a project starts?

I think partly it's an instinct that you gain from experience. The more things you see that go wrong and the more you take the opportunity to learn from the failures, the more you know what red flags to look for. The key is keeping your cycle short so that you have a chance to see them sooner—you need to bring feedback in at a frequent rate.

Data scientists tend to get excited about their work and forget to pull their heads up. It's really important to have not only an understanding of where you want to go at the end of the day, but also what success looks like at different points along the way. That way, you can check your work against that, get feedback, and be able to pivot if necessary. Checkpoints let you quickly know when you've missed or misunderstood something and correct course, rather than learning that at the end and having to backtrack.

13.4.3 How does the way a failure is handled differ between companies?

It's highly tied to the culture of the company. I would advise folks when job searching to try to find out if the company has a culture of learning and ongoing feedback. When you're the interviewing, you have an opportunity to ask the interviewer: What are you learning yourself? How did that opportunity arise for you? If I were to take this role how would I get feedback? Is it something that you have to seek out, or is it formalized? Getting the feel for how employees respond to these questions is very, very telling.

When you're already at a company, there are questions you can try to answer for yourself to see if there's a healthy culture. After a project is finished, is there an opportunity to pause and look back? Do you try to retrospectively learn at the end of projects? Do you see the leadership use open communication and taking ownership for failures at various levels in the company? You get a sense for fear too when a strong culture isn't there. You start to see behaviors that are more self-serving than mission-serving, and that unhealthy behavior kind of just hits you.

13.4.4 How can you tell if a project you're on is failing?

You can't know if you're failing if you haven't from the outset defined what success is. What are the objectives that you're trying to attain, and what do checkpoints along the way towards that success look like? If you don't know this, you're just taking a stab at if the project is going well or not. To set yourself up for success, you have to make sure you've collaborated with your stakeholders to have a well-defined answer to those questions. You need to know why you're doing this project and what problem you're trying to solve, or you won't know the value of what you're delivering and if your approach is correct. Part of the role of a data

scientist is to bring your expertise to the table and help frame the problem and define the success metrics.

13.4.5 How can you get over a fear of failing?

You need to remember that you actually want some failures—if everything went perfectly, you'd never learn anything. How would you ever grow? Those experiences are necessary, as there's no replacement for dealing with failure and becoming okay with it. It's true that failure can be painful, and you may ask, "Oh, my gosh, what am I going to do?" But after you see yourself bounce back, learn from it, and turn it into the next thing, that resiliency you gain will snowball into confidence. If you know to expect some things to go wrong, that makes it easier the next time around. And if you make sure you're frequently getting feedback, you're going to catch your failures before they become devastating. No one expects perfection. What is expected is for you to be honest about what you don't know and to keep learning by asking questions and looking for feedback.

13.5 Summary

- Data science projects usually fail because of inadequate data, a lack of signal, or not being right for the customer
- After a project fails, catalog why and consider pivoting or ending it
- A project failing isn't a reflection on the quality of the data scientist
- A data scientist isn't solely responsible for a project failing

14

Joining the Data Science Community

This chapter covers

- Growing your portfolio of projects and blog posts
- Finding and getting the most out of conferences
- Giving a great data science talk at a meetup or conference
- Contributing to open source

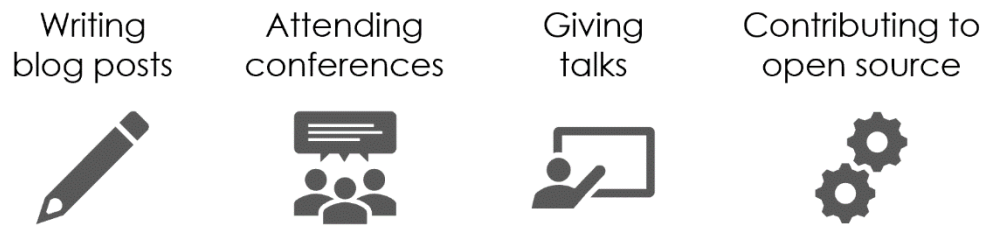
When you're in a data science job, doing well at it can feel like the only way to advance your career. But there are many other ways to grow your skills, especially by engaging with the data science community. Spending time outside of work doing activities like presenting talks or contributing to open source can be enormously beneficial for your career.

In this chapter, we'll be going over four ways you can join the community: growing your portfolio, attending conferences, giving talks, and contributing to open source. We present four activities so that you may choose which ones you like best—very few people have the time and energy to do them all. While this will take time outside your normal day job, that doesn't mean it should take over your life. Throughout this chapter we'll give advice about how to use your time effectively with tactics like reusing talks, turning a blog post into a talk, and writing a blog post about your first open source contribution.

While these actions can be helpful and immensely rewarding, you don't have to do any of these to have a fulfilling data science career. Plenty of data scientists, including those in senior positions at top companies, don't do any of these. But both of us authors feel that being a part of the community has helped us at numerous times during our careers, including in getting job

offers and getting promoted. Doing public work is an area where the time you invest can be paid back doubly. We see four main benefits of joining the broader data science community:

- **Gaining skills:** By engaging with the community, you'll learn new techniques that you wouldn't be exposed to if you only did your day job. Creating an open source project is the activity that will most directly develop your technical skills since you'll be writing code for others to use and working collaboratively on a technical project, but every activity has its benefits. Blogging is a great way to realize gaps in your knowledge and get feedback. Giving talks helps hone your presentation skills that may allow you to convince a stakeholder you need funding or that they should back your project. Hearing the right talk at a conference could unblock an important project and save you hours of work.
- **Growing your network:** Connecting with the community is a great way to find a supportive group of peers who understand your struggles. Even if you have peers at your company, you may be lacking expertise in a certain niche that can be filled with advice from a community member. You can also learn what's it like working in data science at different companies.
- **Gaining opportunities:** The more involved you are in the community, the more you'll be asked to help on projects, give talks, or speak on a podcast. You may even land your next job thanks to someone finding you through your online work or meeting you at a conference. This is a large positive feedback loop: talks lead to more talks and projects lead to more projects. These opportunities can be informative, interesting, and fun.
- **Giving back:** This is less of a direct benefit to yourself, but integral to the well-being of the community. When you ask a mentor how you can pay them back for their support, many will say "Pay it forward. Help others and become a mentor to them." Being a part of the community can make a data science job much more fulfilling. By doing tasks that help others, you'll feel valuable and like you're doing work for more than just a paycheck.



Some ways to join the community covered in this chapter

14.1 Growing your portfolio

Just because you now have a job doesn't mean you can forget all of the excellent habits you developed to get it. In Chapter 4, you learned about writing a blog and building a portfolio. Being employed doesn't mean there's no value from continuing to maintain and expand them. Working on a blog or side project also doesn't have to be burdensome: we'll discuss new topics and ways to recycle your work to get multiple uses from the same effort.

14.1.1 More blog posts

You are hopefully learning a lot of new things in your job as a data scientist. How can you optimize SQL queries on a 30 billion row table? How do you work effectively with marketers? What are some strategies to start navigating hundreds of tables?

If you're at a company with other data scientists, you'll be learning directly from them, whether by reading their code or pair programming. It's a good idea to take notes during this process, as you'll be getting a lot of new information and it's unlikely you'll remember it all in a few months. Once you're doing that, why not share your notes with the class (in this case, the class is the entire internet)? And strangers on the internet aren't the only ones who will benefit - writing a blog post is a great way to consolidate your learning. You may even find yourself years later referring to an earlier tutorial you wrote.

If you followed our advice in chapter 4, you should already have a blog that is up and running with a few posts. If you didn't but you're interested in starting one, we recommend going back to that chapter and following the steps there. Everything we wrote there still applies – the same strategies that made effective blog posts when you were looking for your first data science job hold up even once you're working in the field. The only major change is that you need to make sure if you're writing about projects you've done at work (vs. general programming, statistical, or people management skills you've learned) that you aren't sharing any confidential or proprietary information and you follow any other rules your company has for employee's personal blogs (like running them by the communications department first).

If you don't want to have your own blog, see if your company has a tech blog. Even if past posts have been engineering-centric, you can do a data science one. It might take time to get approval, but a bonus of going this route is that you can write up your thoughts during work time. Even if your company doesn't have a public blog, they should have some internal documentation and training. If you find you had to learn something by asking multiple people or wading through outdated instructions, create or update a resource with clear instructions for future new hires. If the topic is something that will be helpful to people outside the company (e.g. not on a propriety internal tool or description of your data), you can later turn it into a blog post or talk.

14.1.2 More projects

Data science projects, which we also went over in chapter 4, are where you pick or create a dataset and analyze it to answer a question. For example, you could use the Twitter API and

make a network analysis of the users tweeting about a data science conference. In some cases, it doesn't even have to be an analysis; maybe you show off your engineering skills by building a Slack bot to allow users to give each other "points," keeping track of the totals in a database you set up.

Projects can be much harder than blog posts to keep up with. Depending on your industry, your company may be open or even encouraging of writing in broad strokes about your work. Even if they're not, you can write non-technical posts about how to deal with business stakeholders or your experience on the job market. Very few companies, however, will share their data publicly. This means even if you could share the code of an awesome analysis you did, it won't make much sense since you can't share the data or the results. If you want to share analyses you do, you'll have to do it as a side project strictly on your own time.

That said, it is good to occasionally do a side project. For one, when you want to move to your next job, companies may ask for an example of a data analysis you've done. If you've been working at your company for a few years, you don't want to show them a project you did while in a bootcamp or enrolled in a MOOC, as you want to showcase how your skills have evolved since working as a data scientist. That said, the principles of how to find a topic and write a good analysis remains the same from chapter 4.

The good news is a project doesn't have to take a lot of time. David Robinson, whom we interviewed in Chapter 4, does a weekly screencast where he records himself doing an analysis on a dataset he's never seen before (from the Tidy Tuesday project). This takes him about an hour as he does no preparation, but when the code is uploaded to GitHub it could serve as an example analysis project. Now, it does take a pretty experienced data scientist to make a good analysis so quickly, but we recommend to anyone using a time limit so you'll keep focused on sharing results.

14.2 Attending Conferences

Sometimes being a part of the community means having to get out of your home. For the most part this means going to conferences, where people in (or wanting to be in) similar industries or fields come together to talk about their work.

Conferences are usually annual events that take place all over the country and the world. There are many different conferences put on in the field of data science. Strata, RStudio::conf, PyData, EARL, and Open Data Science Conference are just a few of the larger ones, and many of them have regional offshoots. You may also be interested in more general technology conferences that overlap with data science, like Write/Speak/Code, PyCon, Grace Hopper, and SciPy.

Conferences tend to range from two to four days long and have programming from early in the morning until the evening—plus social activities after. They may be single track (only one talk happening at a given time) or multi-track, but they all have multiple speakers. They also can be very expensive, usually between $300 and $700 per day just for the ticket. Some

conferences will also have half-day to two-day workshops beforehand for an extra fee, usually around $750 a day.

One reason we gave approximate prices is that there are many ways to pay less than full price for a ticket. If you're a member of an under-represented group, look for scholarships or discount codes that are offered to all members of a group like R-Ladies or PyLadies. If you work at a non-profit or in academia, you may also pay less, and many major conferences will give discounts for purchasing in advance. Another great way to lower the expense is to give a talk, as conferences should give you a free ticket. Finally, some conferences will have scholarships that you can apply to that will cover your ticket and potentially your entire cost, including transportation and accommodation.

Given this potential steep price, why should you spend your time and money (or your employer's money) attending, especially if the conference records and puts the talks online? Here we'll return to one of the primary benefits we talked to at the top of the chapter: networking. Networking can have a negative connotation of someone going around a room, glad-handling everyone in hopes of meeting an "important" person and gaining something from them. But networking at its best is about finding a community of people who support you. That may be in very tangible ways, like introducing you to someone who works at a company you're applying to, or intangible, like the feeling of finally being in a room of technologists where the vast majority are women. Building a network is best done over the long-term, so even if you don't have anything you think you need help with now, it's great to lay that groundwork before you're looking for a new job or a partner for an open-source project.

> **DRESS CODE** A common question of first-time attendees is what to wear. In general, conferences are going to be casual dress. For a specific conference, see if you can find pictures of the conference on twitter or the conference website. One thing to keep in mind though is that the speakers may be more formally dressed than the audience; just because all speakers are in business casual doesn't mean the audience is too. If you're really stuck, bring things that can fall in the middle, like a casual dress or a button down and dark jeans. It's rare that a conference will be so on the extreme you need to either be in a suit or a t-shirt and shorts to fit in, and there will usually be at least some range in people's dress, so it's unlikely you'll be an outlier!

With some many conferences to choose from, how do you figure out which ones are worth attending? There are many ways conferences vary, but here are a few axes to consider:

- **Academic:** Some conferences, like useR!, NeurIPS, and JSM, have a large number of academic attendees or in research-heavy industry jobs. On the extreme end, there are some conferences where essentially everyone is a graduate student or professor. If you're in industry, you may not find as many of the talks applicable; while people in industry may be presenting, it might be on a cutting-edge machine learning algorithm that's only useful if you're a giant e-commerce company.
- **Size:** Conferences can range anywhere from 150 to tens of thousands of people. We recommend starting out with a small to medium size conference, between 200 and a

thousand people. The smaller size means it's less intimidating to navigate and you're more likely to run into people multiple times, leading to stronger connections.

- **Hiring companies:** On the other hand, one reason you may want to go to a large conference is if you're looking for a job. While you can meet someone who's hiring at any conference, some larger conferences have job fairs where employers pay specifically to set up a booth and talk to potential employees.
- **Level of talks:** Most conferences will generally for people working or studying in the field. For example, if you don't know any R, you probably wouldn't get much out of RStudio::conf, a conference run by the company that develops the primary IDE (interactive development environment) for R. Conference talks are generally aimed at an intermediate level of general knowledge, but vary in the level expected for specific knowledge. For example, RStudio::conf could have a talk introducing a package for time series. You would need to know some R to understand the talk, but the speaker wouldn't expect you've worked with time series much. Or a talk at a conference about online experimentation might be an introduction to how qualitive research can complement quantitative methods.
- **Diversity and inclusivity:** Unfortunately, not all organizers are concerned with making sure their conferences are welcoming to all. If you see all 45 speakers are men, you can take a good guess that the attendee population is going to look similar. Besides the speaker line-up, look on the website to see if there's a code of conduct. If you need certain accommodations, like having the venue be wheelchair accessible, look for an email address on the conference website and ask.
- **Specialty:** Just as data science has many specialties, so do conferences. Whether you're looking for a specific language or domain, there's probably a conference right for you.

Once you've decided what type of conference you're interested in, look for reviews of conferences before you commit to going to them. If you don't know anyone personally who has been, ask on Twitter or LinkedIn. Also look at the schedule for the conference or, if it's not available yet, the schedule from the previous year. If there's recordings of a talk, watch a few of them. You want to make sure your investment is worthwhile and unfortunately, some conferences do not have many good speakers.

Going to these conferences can be helpful for your career and also for your employer, who gets to have you representing them and learning things that can help you do your job better. That means that you may be able to convince your company to pay in full or in part for the conference. Some companies have a formal conference or training budget. That's great in that there is already money earmarked for your purpose, but if you want to exceed the budget it will be difficult to get them to make an exception for you.

Beyond the cost, most conferences happen at least partially during the week, so you'll have to take a day or two off of work. You don't want that coming out of your vacation days, right? You will need to make the case to your manager that it's worthwhile for you to spend

the day at the conference rather than doing your normal work. Some companies have a policy, which may or may not be formal, of how many conference days you can take. At tech companies, it's pretty normal to at least attend one conference, but other industries may not have that system.

If you need to make the case for conferences to your manager, here are some benefits to focus on:

- **Recruiting**: It costs thousands or even tens of thousands of dollars to recruit a data scientist. One of the biggest problems is getting good candidates to apply in the first place. Huge tech companies like Google and Amazon and hot start-ups may have great candidates knocking down their door, but most companies don't have that name recognition. If you meet people at the conferences, it puts your company's name out there. This is magnified a lot if you're speaking, which we'll be covering later on.
- **Knowledge**: Your manager wants to know what you will be able to do after a conference that you couldn't do before. It's even better if you share that knowledge with the team by writing up an article (which can also become a blog post!) or presentation. Start by looking at the conference schedule and reporting back to your manager the talks that will be immediately applicable in solving your problems. But, keep in mind that conferences also have the "hallway track": the informal conversations happening outside of the presentations - you may find someone there who has the solution to a problem you're facing! Five or ten minutes of the right person's time can pay for the cost of your ticket.

If you live in or near a big city, look for a conference there so you won't have to pay for travel or accommodation. Show the value you've gotten out of that by connecting to business wins or impact, will help make the case for a larger investment.

14.2.1 Dealing with social anxiety

It's a cliché that engineers and scientists are socially awkward introverts, but most people struggle at some points with social anxiety. Even the most confident people don't walk into a roomful of strangers and feel totally comfortable. What should you do if you're nervous you'll get to a conference and just hide in the corner looking at your phone the whole time?

Fortunately, the advantage of attending a meetup or conference is you have something built-in to talk about! In general, a solid strategy is to ask questions – people love to talk about themselves. You can ask why they're at the talk, how long they've been programming in X, or whether they've been to this meetup before. If it's a conference, inquire if there are any talks they're especially excited about. Remember that many people are feeling awkward – it's not just you. If you're nervous, a nice time to try talking to people is in the few minutes before a talk when you're in your seat. Sit next to someone and strike up a conversation – if your fears about a bad conversation are realized, you know it can only last a few minutes since the talk is about to start!

When you're in a room with a lot of people, look for those standing in a "pac-man" shape: a circle with an opening. Head over to stand in the opening and try to shift around so an opening reemerges for more to join. You don't need to introduce yourself as soon as you arrive; you can wait for a break in the conversation or even just join in the conversation without an introduction, especially if it's a large group.

We've talked about imposter syndrome before, and this is another area it can strike. You might end up at a talk that goes way over your head. The biggest thing to remember is that you should not be made to feel like an imposter. If people treat you like you're not worth their time, or act shocked if you don't know a term, or make a belittling comment, that's on them. There are meetup groups and conferences that will be welcoming. Many people love helping others and remember what it was like to be new. If you end up at a meetup or a conference you don't enjoy, try not to let that dissuade you from ever going to one again.

While we've covered some strategies for meeting people, it's totally normal to take personal time during the conference. Several days of socializing with strangers can be exhausting. It's a common mistake to feel that you need to be doing something "productive" every moment of the conference, whether that's attending a talk or networking. But you absolutely do not! Don't feel bad about taking a walk by yourself instead of attending a talk during one session; you'll get more out of a conference overall if you take time to recharge.

14.3 Giving talks

Giving talks can offer you a lot of opportunities for growth, and it will also give you leverage to go to more talks and conferences. One fight that you might have is to try to get enough time to go to conferences to improve your skills and network, but giving talks is a great way to visibly represent your company (in addition to the monetary benefits of talking which makes it less burdensome to your employer). While it may seem like you need to be an industry expert or a spectacular orator or social butterfly to give a talk, that's not the case. Giving talks is actually a great strategy for an introvert. After your talk, people will come up to you to give compliments, ask follow-up questions, or just introduce themselves. It's an enhanced version of the general advantage of having the topic of the meetup or conference to talk about.

This section could be a whole book in itself, and in fact we have recommended in the resources section some books on public speaking. We want to emphasize that the bar for giving a good talk is lower than you might think. You're not giving a TED talk here or keynoting of a huge conference. Those folks have tons of experience and have likely hired speaking coaches. We believe if you want to give a good talk you should just focus on two things: entertaining people and motivating them. If people aren't interested while you're speaking, it will be very hard to teach them anything. Additionally, there's a limited number of take-aways people can get from a twenty, thirty, or even sixty-minute talk. But if you can fan the desire to learn more and equip the audience with how to get started, you'll have delivered a lot of value.

14.3.1 Getting an opportunity

How do you find opportunities to speak? The best place to start is looking for conferences that have "calls for proposals," or CFPs. This means you can submit a short summary of your talk, called an abstract, and the conference organizers will pick from the submitted talks who they want to speak. Some conferences do "blind" reviews, where they pick abstract without knowing anything about the speakers, while others will ask to know more about you.

When you're looking at conferences to speak at, apply similar criteria to when you are looking for ones to attend. If going to a conference with 10,000 people sounds like your nightmare, you probably don't want to apply to speak there. Moreover, speaking is a great way to lower the cost of attending a conference, so why not get that benefit for one where you really want to see the other talks? Make sure to ask people you've met online or at meetups what are conferences they recommend – great, smaller conference are easy to lose under the radar.

The first part of a good abstract is paying attention to what the conference is asking for. Even if you write the best 500-word abstract, you're not going to get in if they were asking for a 150-word one. Ditto if you submit a talk on data engineering to a conference only focused on statistics.

Overall, a good abstract has a first sentence that's a hook: it draws the reader in to learn more. You then should explain what problem you're solving for the audience and give an overview of what they'll learn. Here's an example from one of our previous talks:

> Deep learning sounds complicated and difficult, but it's really not. Thanks to packages like Keras, you can get started with only a few lines of code. Once you understand the basic concepts, you will able to use deep learning to make AI-generated humorous content! In this talk I'll give an introduction to deep learning by showing how you can use it to make a model that generates weird pet names like Shurper, Tunkin Pike, and Jack Odins. If you understand how to do a linear regression, you can understand how to create fun deep learning projects.

When coming up with an abstract idea, a great place to start is by thinking of the you of three months, six months, or a year ago. What do you know now that you wish you knew then? It's easy to feel like everyone already knows this things, but even for stuff you consider "basic," like how to use git and GitHub or how to do web-scraping, there are thousands of people out there who don't know it and would benefit from it. You can also choose to go into your sub-field if it would be interesting to a broad audience; for example, maybe you can teach how to make interactive maps, or use a package tailored to fast data analysis, or what a generalized linear model is. You don't need to be the "expert" in the field – in fact, people who've just learned something are often the best teachers. Those who've learned it a long time ago forget what it was like to struggle and what misconceptions they had.

Another great way to start giving talks is speaking at local meetups. See if any are hosting lightning talk events, which are a series of short (often five-minute) talks. It's a lot less pressure to prepare as they'll have anywhere from five to a dozen speakers in one evening. These usually are explicitly welcoming to first-time speakers. If there's no lightning talk evenings planned but there's a local meetup you faithfully attend, recommend a lightning talk event to the organizers!

Starting R-Ladies by Gabriela de Queiroz

When I moved to San Francisco from Brazil in 2012, I was amazed by the number of resources I found. I quickly discovered the meetup scene and for a few months, I would go to meetups every night. Learn and eat for free: it was a perfect combination, especially for a student who didn't have much money. But the majority of meetups didn't have a diverse audience. I didn't see anyone like me and couldn't feel welcome so I would end up in a corner, not interacting much.

After a while, I decided that it was time to give back to the community and start my own meetup. I was passionate about R, but I didn't want to create a regular R group; I wanted a group where I (and the attendees) could feel safe and welcome, with no judgment, and we could see ourselves in the audience. That's how R-Ladies was born. In October of 2012, I hosted the first event, an introduction to R, and only 8 people showed up. I was a little disappointed, but I was happy to be creating this space and for being brave enough to teach a programming language in a foreign language.

For 4 years, I was the only person behind R-Ladies. I was organizing, hosting, teaching, advertising, running the website, and looking for places and sponsors. I would go to conferences and events and talk about the group, I was active in social networks trying to make as many connections as I could. Unfortunately, most of my employers would not sponsor my work, so R-Ladies was my side project, which means I would spend nights and weekends working on it. Leading R-Ladies gave me the opportunity to meet numerous people, some of who I would never have dreamed of meeting in real life. And because I had to teach in the events, I became more comfortable speaking in front of people. For people wanting to start their own communities, I would suggest a few things:

1. **Define a purpose and create a mission statement**
   What is the goal of this community? What are you trying to achieve? Why are you creating it? What is the mission of the community? Who will be the audience? Thinking about the questions above will help your future members understand the reason why they should care and why they should join. It also helps influence decisions like whether you want to focus on a specific subgroup, like R-Ladies did with women and gender minorities, or whether you want to reach everyone interested in the topic.
2. **Set up social channels, a website, and an email**
   Set up a Twitter account, Facebook page, LinkedIn group, Instagram profile and any other social channel that has a big userbase. You'll also want to have a website and email so people can easily contact you and find out more about the group.
3. **Create a logo**
   Having a logo brings awareness of your brand and therefore your community. Some people have a better visual memory and they will remember your logo. With a logo, you can make laptop stickers for example. Laptop stickers are a way to express yourself, your beliefs and the communities you are part of. It is a big hit!
4. **Think about the format**
   Is this going to be mainly talks or workshops? Will it all be in person, or will you be an online community with live-streamed events or "coffee chats"? If your community is a tech community where you want to empower your audience, a workshop would be a great format. Active learning is the best way to learn something.
5. **Use a platform** (meetup.com or eventbrite.com for example)

You want to make it easy for people to find and register for your events. A centralized website like meetup.com or eventbrite.com will allow for some organic traffic when people are searching for the topic and help keep track of your expected attendance.

Building a community requires time and effort. You will probably need to work after hours and on the weekends so make sure it is something that you are passionate about and whose mission you believe. Despite the work, it is worth it. Hearing the success stories, seeing how your community changed local communities around the world, especially in underserved places, is very rewarding and a source of great joy. You feel that you are doing something to change the world for the better. Good luck in your journey!

Finally, you may be able to get people to come to you through your blog. For conferences that have invited speakers, if one of the organizers reads a blog post that fits perfectly with the theme of the conference, they may reach out to you to see if you could design a talk on the same topic. Even if this doesn't happen, blog posts aren't great ways to show conference organizers that you are effective at communicating when you don't have previous talks to point to.

Just like data science jobs, getting your first speaking gig is the hardest. After that, you'll often experience a snowball effect, especially if your talk was recorded. A recording is great because people may see your talk and reach out to you, but also because some CFPs will ask for a recording of a previous talk.

14.3.2 Preparing

Once you've got a speaking engagement, you'll want to spend a lot of time preparing your talk. If you've never given a public talk before, it's easy to under-estimate how much time it takes. Yes, you could throw together a talk at the last-minute by making every slide five bullet points that are just your thoughts on the subject matter and wing it the day of, but this is disrespectful to your audience and not showcasing you at your best. It's also not the road to building a successful speaking side-gig.

You want to practice giving the talk to a live person, not just reading the slides to yourself. Find someone whose criticism you trust and give the talk to them. Unless you are giving a very technical talk, it probably doesn't matter whether your reviewer has a background in your subject. They can tell you the general things that will make your talk better like if you use lots of filler words or move your arms around a lot.

You will generally be given a timeframe for your talk, but it can be complicated based on whether there will be a Q&A after. To calculate how long you need to prepare for, you should generally budget five minutes for questions and work backwards, but it's a good idea to time yourself giving the talk. Be careful as there is a temptation to speed up when giving it in front of people. You might also want to add some extra slides at the back for "if we have extra time" as you might find you hurried a little too much through the main talk. If you're under by a few minutes though, it will generally be fine and there will just be a slightly longer break

before the next talk. The worse outcome is that your talk is way too long and you either get cut off before finishing or finish over time, disrupting the next speaker's slot.

Because of all the work that goes into any talk, we highly recommend re-using talks. It's very unlikely there will be overlap in the audience, especially if the talks are in different cities or given at a multi-track conference (where attendees could choose to go to another talk that's happening at the same time). While it would be flattering to think everyone has watched the recording of your talk, most people won't have.

The day of, gather your supporters. This doesn't need to be limited to your friends and colleagues in the data science space; invite your family, partner, that friendly person in your building. If it's a paid event, see if they'd give a pass to a family member or partner to just attend your talk; Emily's grandfather has been to multiple of her talks for free (much to the delight of the other audience members). It's nice to know that at least part of the audience is definitely in your corner.

14.4 Contributing to open source

For those of you who like the idea of being part of a community but dislike the thought of standing in a room with other people in that community, open source can meet that need. Contributing to open source allows for sharing of ideas and develops a sense of community among people with the same passion. Creating a project in open source can generate lots of interest as people will push it in new directions that you possibly hadn't considered. Similarly, you might be able to expand upon someone's work to generate an entirely new project.

R and Python are able to thrive because of volunteers who continually expand and refine them. We'll be discussing in this section how you can become one of those volunteers, but you can also contribute financially to the organizations that sponsor some of the core development. While R and Python may be free to use, they're not free to maintain and develop. The R Foundation, the Python Software Foundation, and NumFOCUS are three organizations (the latter two registered US charities) that you can donate to in order to support the continued development of the languages.

14.4.1 Contributing to other people's work

Hopping onto an open source project can feel like peeking into someone's closet. It's their space and you feel like an intruder, but open source was built for exactly this purpose and you have to get past this feeling. Instead, imagine open source projects are like throwing a giant dinner party. You probably don't want to be responsible for making the main course just yet, but there are lots of jobs that need to be done - you can help set the table, make sure everyone has water, or put the dishes away afterward. If you're respectful and enthusiastic, most creators and maintainers will welcome your help.

A great place to start contributing is documentation. See how fleshed out the documentation is for a package you like. You might see something that is incomplete, unclear

or misleading. Even a typo fix is worth making a pull request on GitHub for. The creators of packages and libraries love to get more work written on it. Not only does it save them time, but as someone who recently learned how to use it, you'll have a better perspective on what will motivate and teach a new user.

If you do want to contribute code, don't jump right in and start re-writing things or submitting a new function. If it's a large project, it may have a guide for how to contribute or a code of conduct. If it doesn't, watch the repository for a while to understand the flow. This will also tell you if it's an actively maintained project or one that is dormant for long periods of time.

Working in open source is one of the best ways to grow your technical skills, especially if there's nothing at work that requires you to cooperate with a large group of people. Maybe on your work GitHub repositories you don't use branches, informative commit messages, or tagging. That's okay, but when you enter a project with hundreds of issues and dozens of people working on it at once, that extra work starts to make more sense. These types of practices do add extra constraints, whether that's working within a style guide or the maintainers not adding a feature you created since it's not performant enough. Ultimately, they'll be the final decision-makers until you create a project of your own. While this can be frustrating, you'll learn a lot of best practices you can apply to your own work.

Hackathons by Reshama Shaikh

Contributing to open source can seem enigmatic and daunting. Open source sprints, sometimes referred to as hackathons, are structured events that provide a welcoming space for beginners. Sprints are typically one or two-day events where participants work on open issues submitted to the GitHub repository of a Python or R library. These issues could be related to documentation, bug fixes, tests, feature requests and more.

The benefits of participating in open source sprints are many:

- The majority of open source contributors are volunteers, so community involvement is essential and welcome.
- It is an active, hands-on event which builds engineering and coding skills.
- Contributing to open source is an excellent learning opportunity which advances one's data science skills and builds your portfolio.
- It provides a valuable networking opportunity with other data scientists and experienced contributors.

A well-organized sprint will utilize people's time effectively. The preparation ensures that beginner contributors are able to leave the sprint having accomplished something. Look for an available central repository of resources and preparation work, which includes the contributing documentation, R or Python installation instructions, tools to sign up for prior to event (like GitHub account or messaging platform), and a curated list of open issues specially prepared for the sprint participants. Keep in mind that people who organize these sprints are volunteers - if you find that some of these items are lacking, offer to help. Organizing open source sprints is also contributing to open source.

The goal of an open source sprint is to submit pull requests (PRs) that resolve open issues. Submitting a PR is a back and forth process and it is common for it to take several weeks to be merged in. Allocate some time post-sprint (typically 5 to 10 hours) to follow-up on work and see a pull request through to "merged" status which is represented in the GitHub repo as a beautiful purple icon.

If you're interested organizing a sprint yourself, I wrote a detailed guide on my blog at https://reshamas.github.io/how-to-organize-a-scikit-learn-sprint/.

14.4.2 Making your own package or library

When you find yourself copying functions between projects or messaging them to your coworkers, it may be time to make a package or library. A package allows yourself to store functions in one place, easily share them, and enforce best practices like testing the code. Many companies have internal packages that have functions to make the color of your plots the corporate color, access data, or solve common problems. If you think others may also be facing the same problem, you can share your package on GitHub so other people can download and use it.

Before you try to get the public to use something, you need to make sure your code is all in order. Just because something has worked well for you running one task doesn't mean that it will hold up under the stress of public consumption. If your code is something that you just cobbled together, but you're not sure how it works, then don't invite people to use it just yet. Making your package more broadly useful may require more advanced programming as you refine or adapt the package to fit a generalized case. Make sure your underlying work has been read through by someone you trust. Users won't look under the hood, so if you tell them it's a Ferrari, they'll be upset when it turns out to be a golf cart half the time.

Once you've tested your code and had it reviewed, you still will have to do work to get people to find out about it. You can market it on social media or your blog and at, but even then it might be slow going. Don't expect to be a star overnight, but it's better to put in work early with fewer users than to be immediately successful and then realize that you made an error in your underlying assumptions. It can take a while for something to be adopted, if it ever is, but even attempting to spread good work is a good deed. Of course, the reward of success is also a curse – if people start relying on your project, it becomes very hard to stop developing it. You'll get bug reports and feature requests and you'll actually have to seriously consider if you're going to make a change that will break reports that used the old version of a function.

TOXICITY IN OPEN SOURCE Open source communities can be toxic. People have had negative experiences where they're discriminated against, harassed, belittled, or just made to feel unwelcome because of their race, gender, ethnicity, or sexuality. Fortunately, many communities are recognizing this and actively working to make the environment more inclusive. For example, Guido van Rossum, the creator of Python, has committed to only mentoring women and underrepresented minorities (https://qz.com/1624252/pythons-creator-thinks-it-has-a-diversity-problem/). Some project creators will tag issues with "beginner-friendly" or "first-time" to encourage those new to open-source to contribute. You should always prioritize your mental and emotional health, but there are many people, including those from underrepresented groups, who have had only positive experiences in open source – a bad one is not inevitable.

14.5 Recognizing and avoiding burnout

We are not health experts, so we defer to the World Health Organization's definition of *burnout*: a "syndrome conceptualized as resulting from chronic workplace stress that has not been successfully managed." It lists the three symptoms as "feelings of energy depletion or exhaustion", "increased mental distance from one's job or feelings of negativism or cynicism related to one's job", and "reduced professional productivity" (https://www.who.int/mental_health/evidence/burn-out/en/). For now, we're focusing on stress that doesn't come from your full-time job but the extra career related work you do on the side.

Writing this book was something we both did completely separate from our full-time jobs (and, for Jacqueline, separate from raising a toddler). We certainly sometimes get jealous when colleagues go home and do nothing data science related. For us, it helps to go back to why we decided to take on this extra work and see if we're still working towards our goals. With this book, it's never been about making money (spending those writing hours consulting instead would have been much more lucrative). Rather, we wanted to write this book to help aspiring and junior data scientists, and that mission has kept us motivated. It was especially helpful to see positive impact along the way as we released chapters.

If you feel like you are burning out, start by asking yourself if there are ways that you can cut back. Something that's helpful to remember is that once you've created something, you don't have to remain as active. For example, if you keep a blog, you may want to occasionally write a new post, but you don't have to write as frequently as you did at the beginning. Someone visiting is more likely to say, "Oh wow, those 6 posts are really helpful to me" than, "Oh, she now only posts once every six months."

In today's culture of "side hustles" and the praise of busyness, it can feel like any time not spent being "productive" is time wasted. This is very damaging! We all need time to reset. Things like going to the gym and hanging out with friends are good for this, but it's also okay to take time to watch television or "veg out." Continue to make time for hobbies that have nothing to do with data science or making money so you don't feel like your whole life revolves around work.

You can add a lot of stress to yourself by trying to "keep up" with people. It's a common saying that social media is someone's highlight reel and you shouldn't compare that to your full life. Similarly, just because someone is a prolific maker of packages or blog posts doesn't mean that you need to keep up with that person. For some people, creating packages, giving talks, or writing blog posts is part of or even the full mandate of their job! The best thing you can do for your career is make sure you're working sustainably enough that you can stay in it for the long-term.

14.6 Interview with Renee Teate, Director of Data Science at HelioCampus

Renee Teate is known on Twitter by her 45,000+ followers as "Data Science Renee" at @becomingdatasci. She's also created a podcast, blog (becomingadatascientist.com), and datasciguide.com, an online data science learning directory. She regularly speaks at and organizes conferences.

14.6.1 What are the main benefits of being on social media?

Twitter has benefited me in so many ways. All of the guests that are on my podcast are actually people that I met through Twitter. I found people I thought were tweeting about interesting things and figured if they can tweet about interesting things, they could also talk about interesting things. I made a list and sent a bunch of direct messages all at once, thinking maybe half of them might be interested and I could schedule in the future. Well, every single one of them said yes!

I'm regularly asked to speak at conferences and meetups through Twitter. When I put out content, I know there's an audience for it and it will get engagement. I've met so many cool people. Besides the networking, I also use it to learn. I actually wrote a blog post early on about how I use Twitter to learn things, and it's mostly focused on picking up the lingo of the industry. If you start following people in a certain industry and reading the articles that they're linking to, you pick up all that terminology. If somebody said something I didn't know, I would just go look it up and figure out what that thing was. Often there would be a link to a tutorial or paper on the topic, so it really helped with learning as well.

14.6.2 What would you say to people who say they don't have the time to engage with the community?

I really understand that, especially for people that have responsibilities outside of work, like caring for a child or other relative. When I was in my master's program and working full-time, I didn't do anything else. In those cases, I would advise to find an online community that you can engage with asynchronously. Whenever you have a little bit of time, like you're in a waiting room somewhere, you can read and respond to some tweets or bookmark interesting articles to read later. Even if you can only make it to one event a year, pick a conference that's related to data science or your particular industry and make an effort to squeeze it in. You can keep up with the people that you meet there afterwards on LinkedIn or other social media, and eventually they can be a small group where you could hold each other accountable for your learning and share resources.

14.6.3 Is there value in producing only a small amount of content, like writing just one blog post?

Absolutely. I think even if you put one blog post out there, it will help solidify the topic in your mind because you learn by helping other people understand something. I have referenced some of my old blog posts for years. When I blog, I try to make it something that's very generally applicable and beneficial to learners, so I can refer back to it multiple times without it becoming quickly outdated. For my podcast, I've only recorded two episodes in the last year and a half. I got busy with life and had started my job at HelioCampus and set the podcast aside, and it was really hard to get back to it once I committed to other things. It's still out there, and I still plan to make more episodes. But I have stopped feeling guilty about taking a long break. I realize that the episodes that are out there still helpful to people, and I can always pick it back up later.

14.6.4 Were you worried the first time you published a blog post or gave a talk?

Yes, I was anxious about putting something out there for the first time because when people look up your name, they're going to find it and associate it with you. Of course it is a little bit nerve wracking. But one thing that I realized is that the kind of blog posts that were already out there that I got value from weren't always the most technically-advanced perfectly-written pieces. I would read blogs that described something that I wanted to learn in a slightly different way than I had heard before, and suddenly the material clicked. There's always going to be somebody that benefits from what you put out there.

I've also learned not to care too much about naysayers. They're going to be out there no matter what. I've seen people make negative comments to people that have been doing data science for years. There are lots of ways to approach an analysis, and maybe one way is better than another for some reasons, but it doesn't mean that your way is not a good way. Sometimes you just have to stop listening to detractors.

14.7 Summary

- There are four ways we recommend for getting involved with the data science community: building your blog and data science portfolio, attending conferences, giving talks, and contributing to open source.
- Remember you don't have to do any community activities to have a successful career; choose what works for you and don't worry about "keeping up" with other people.

15

Leaving your job gracefully

This chapter covers

- Deciding when to leave a job
- How the job search differs from your first data science job
- Giving notice and managing the transition

The days of spending 40 years at one company and retiring with a golden watch and a pension are over. In most fields, it's now common to switch companies at least a few times in your career, and in tech, people can switch jobs as often as every couple of years. There are many good reasons for leaving a job: you could be looking for a compensation bump, different responsibilities, accelerated learning, or simply something novel. Deciding that you might be interested in a new job is the first step, but there are additional mental hurdles to overcome between that step and actually doing something about it.

There's always going to be uncertainty when leaving something you know for a new role. No matter how much research you do or how many questions you ask in the interview, you can never know what it will truly be like until you start the job. You can understand the big things: what's the salary, the company size, and the structure of the data team, but you won't know how you'll feel day-to-day until you're living it. Moreover, it's likely that your current job isn't completely horrible (if it is, we recommend going back to chapter 9 where we discuss what to do if the work is terrible or the environment is toxic). You probably like some of your coworkers, know where to get help, and are comfortable navigating around the data. You can think of some things that might be better, but what's the guarantee a new job that you think will have that actually will or won't have something else even worse? Is it really worth the risk and time to find a new job?

These nagging doubts can slow your job-hunt progress even once you decide you want to leave. But your search for a second (or third, or fourth) job doesn't have to be intimidating if

you know that simply looking for greener pastures doesn't stop you from changing your mind. You're also probably dealing with a lot of uncertainty. How should you approach your second data science job search differently? If you get an offer you want to take, how do you tell your manager? Do you meet individually with every coworker you've worked on a project with to tell them you're leaving? If you're given a counteroffer, should you accept? What should you do in those last few weeks after you've given notice?

We've raised lots of questions that we ask ourselves as we consider looking for a new job, and the sheer volume of uncertainty can paralyze all but the hardiest of job seekers. Fear not, however, as we are here to turn you into one of these. In this chapter, we divide leaving gracefully into three parts: the decision to leave, the job search, and giving notice. Some of this guidance would apply to any job, but we also discuss some things more unique to data science. Changing data science jobs is a common and usually rewarding experience – many people change every 1 to 3 years as it can allow you to try new areas of data science and drastically increase your salary and other benefits. This chapter will help you make that transition as easy and stress-free as possible.

15.1 Deciding to leave

Unfortunately, most of the time you won't know when it's right to leave with 100% certainly. There's no magic eight ball that will tell you what to do, or even a set of questions you can answer that will give you a definitive decision. In chapter 8 we discussed choosing between two good life choices when deciding between offers, and that same style of reasoning applies here. At the end of the day, you can only do the best with the information that you have, and very few decisions are completely irreversible. You can always still switch from that place too: you're not signing a 100-year contract.

15.1.1 Take stock of your learning progress

What should signal that it's time to look for a new job? Our biggest piece of advice is to make sure you're always learning. Unfortunately, it can be common for your learning to slow down as you stay in one role. When you're in your first few months, you're drinking from the firehose. It's pretty much impossible to not learn anything; at the very least, you're learning about the company's data, new technical skills from colleagues, and working with business stakeholders. But if you keep doing the same thing after a year or two, you might plateau.

As you get more comfortable with the day-to-day of your job, see what you can do to find ways to improve your non-technical skills. See if you can take charge of a team (or, at least, an intern) and work on your management ability. While the work that the company needs you to do may be limiting, as you get experienced in it, you can usually find more time to broaden your portfolio. Maybe you could collaborate with the data engineering team to learn how to build some of the pipeline yourself, instead of relying solely on them. Taking the initiative to push yourself in this way isn't for everyone though, and sometimes people need to by

externally motivated by their company to take on new challenges. If you find yourself in a rut and unable to get out of it, that's a sign that it might be time for a change of scenery.

An exciting thing about data science is that there's always more to learn, but that also makes it challenging. If you don't grow, it will be harder to find a next position. It's also expected a senior data scientist to have noticeably different skills than a junior data scientist, both in breadth and depth. We've emphasized throughout this book that you don't need and can't know everything under the umbrella of "data science," but is expected that you know more as you gain experience.

15.1.2 Check your alignment with your manager

Before you cut ties and run though, make sure you've done what you can to tell your manager what you'd like to change. What may seem to you like an unsolvable problem they could come up with a solution. For example, maybe you're bogged down with doing rote tasks that can't be automated but aren't challenging any more. Your manager may say that you can hire an intern who can do that. That intern gets a learning experience and you get the work somewhat off your plate and experience mentoring. Or maybe the data science team does mostly analytics work, and you really want to start doing production machine learning. Your manager might set you up to "bootcamp" with an engineering team for a few months, where you learn some fundamentals of engineering while contributing your analytical knowledge.

Another question to ask yourself is how aligned your career goals are with your manager. Philip Guo, an assistant professor of cognitive sciences at UC San Diego, wrote a blog post called Whose Critical Path Are You On? where he discusses the importance of knowing your boss's (or mentor's) critical path and whether it aligns with yours. Critical path means "the path of work that's critical for their career advancement or fulfillment at the given moment in time." It's about their success being tied to yours. Managers have limited time and energy and your critical path overlapping means it more likely they'll be able to focus on you.

This requires knowing what your own career goals are. We're not talking about the 10- or even 5-year plan; in such a new and quickly developing field, it's impossible to know what opportunities will be available. But how do you want to spend the next few years? This is something you hopefully thought a lot about in your first job search, but maybe things have changed. You might have wanted to be on a large data science team, but now you find a few years in you'd like to be able to work on different types of projects rather than being siloed. Or you could be focusing on your family and looking for a job where you can do more 9-5 rather than a start-up that will demand a lot of hours.

In summary, some key factors to consider when thinking about finding a new job include:

- Are you learning in your current role?
- Have you tried to improve your day-to-day by discussing your issues with your manager?
- Is your manager focused on your needs and advancing your career?
- Have you spent time thinking about both what you do and do not want in your next job?

Leaving your job without another one lined up

You may want to take a significant amount of time off between jobs. Most new employers will want you to start as soon as possible; while you can generally get a week or two weeks between leaving your current job and starting a new one, especially if you already have a vacation planned, it's unlikely you'll get more. If you've been dreaming of a three-month backpacking trip through Asia, you'll probably need to go without something else lined up.

Leaving without another job is risky. There's the financial risk that you might not have enough savings for an indeterminate amount of time without income. Are you willing to (or able to) rely on short-term loans from family or money from a partner's income? The other part is that it's easier to find a job when you're currently employed. Some of that is unfair stigma by hiring managers against those who are unemployed. Some is that your negotiating position is weaker: whatever they're offering will be more than you're making, so it's harder to ask for higher compensation. Some is that if you have taken months off, you may not have kept your skills up and could be rusty for technical interviews.

If you do want to take time off, it's hugely helpful to have a strong data science network: people who are familiar with your work and can get you in the door and talking to a hiring manager. You also want to plan for some time before interviewing to brush up on your technical skills. Overall, unless you're in a toxic work environment, we recommend not leaving a job without another unless you have a plan for the months you're excited about.

15.2 How the job search differs after your first job

A lot of the basics of finding your second data science job are the same as the first. But you do have some significant advantages now that you have experience working in data science:

- You'll get more (any) recruiters coming to you, especially if you turn "career interests" in your profile on LinkedIn (and don't worry, your employers don't see this).
- You've learned more about what aspects you like and what you don't. You're early enough you can pivot your specialty: if you did a lot of data engineering work but didn't enjoy it, you could go to a bigger company where there are data engineers to do that.
- It will be easier to get to the first hiring screen. Many employers use whether someone has held the same title (or something very similar before) as a quick screening tool.
- Your network in data science is hopefully more developed (if it's not, go back to the previous chapter). If you're still employed, you likely don't want to post on LinkedIn or Twitter that you're looking for a new job, but you can start quietly putting out feelers and letting a few trusted folks know that you're looking. They may be able to refer you to a job at their company or they could connect you with someone they know is hiring.

Don't be afraid to apply to jobs, even if you are reasonably happy at your current one. There are plenty of reasons you can talk yourself out of a change. Maybe you worry that you might not have the skills that you think everyone has: "What if I have years of experience and I don't pass the technical take home?" This is just impostor syndrome talking (and if you have learned anything from this book, let it be to fight that nagging voice telling you that you aren't as good as everyone else). If you don't pass a technical screen, it doesn't mean you're a failure or a "fake" data scientist. There are plenty of bad interview questions out there that

don't accurately judge ability. Data science is also so broad that maybe the questions were in an area that you haven't worked in before.

You might also be you're worried about the repercussions on your social life with your work friends or that a new job could take you farther from home. But whatever your concerns, you'll be doing a disservice to yourself to not be open to the application process that might improve your career.

15.2.1 Deciding what you want

The first step in your job search is to make a list of what you've enjoyed in your current job. One book, "Designing your Life" by Bill Burnett and Dave Evans, suggests making a note before and after every activity for a week of how much you thought you would and how much you ended up enjoying it. Do you hate having meetings a couple hours each day? Or do you actually like it because it brings structure to your day? If you're in a distributed data science team, do you wish you reported about to a data science manager? You can use this list to design your search—find a company where they value the same things you or have the structure you're looking for. You don't want to apply to a company where you face the same issues that are causing you unhappiness now.

In your search, you may run into the title issue. In Chapter 5 we talked about not concerning yourself with titles. Data science is done by people with a lot of different titles, including data analyst, research scientist, machine learning engineer, or product analyst. Data analyst is the most common one and that can be seen as a junior role. If you're a data scientist, should you be willing to accept a "senior data analyst" title? If you're a data analyst, should you concentrate on "moving up" to the data scientist title with your new role?

Learning is still the most important in your search: what will you actually be doing in your role? Think in terms of 5 years, not the next 2. What will set you up for success in the long-term? For example, is it possible to join as a senior data analyst and then transition to a data scientist role? Will working at a smaller technology company let you learn how to work with web data, setting you up to then go onto a large tech company?

When you're considering your options, you do have to protect your market value. Fairly or not, data scientist is still usually seen as the more "prestigious" title to data analyst, and a senior data analyst may pay below a data scientist role. You'll want to balance these different considerations as you think about your next role.

15.2.2 Interviewing

After you start applying to companies and getting interviews, you will now have to face the question, "Why are you leaving your job?" If you got your current job out of school or a bootcamp, you didn't get asked that at all. A good answer is you're looking for challenges. Another good strategy is to pretend the question is "Why do you want to work for us?" If you imagine that, your answer will be positive, like "I've heard amazing things about your machine-learning team and am eager to learn," rather than negative, "my past boss insisted on displaying our experiment results in pie charts." If you give a more specific answer, make

sure it's something that fits the new employer. You don't want to say "I'm looking for a team with senior data scientists I can work with" if the interviewing company doesn't have any! Avoid badmouthing your current job at all costs; some employers view this as disqualifying behavior, no matter how poorly you are treated at your company.

Just because you are leaving your current company doesn't mean you shouldn't be proud of the work you've done there. You should absolutely talk about projects that you've worked on or skills you've learned. You are likely bound by certain confidentiality agreements, so you can't literally show them your code or talk about the parameters of the recommendation algorithm you built, but you should discuss your contributions in a general way. "I created a chatbot in Python that generated responses to frequent client questions, decreasing the average time a customer service representative needed to spend with each client by 5 minutes and increasing customer satisfaction by 20%" is a fine non-specific answer. On the other hand, if you work at a private company, "I conducted A/B tests that brought the company's total revenue from 20 Million to 23 Million," is a bad answer as you're disclosing private financial information.

You may be looking at jobs that use different technologies, whether a different cloud provider, SQL dialect, or main programming language. Here, you want to use the similar strategies to when you framed your previous work experiences in terms of skills transferable to data science. For example, let's say you've been working in R and they use Python. You could say something like, "I know it will take a little time for me to get up to speed on the syntax in Python, which I've already started doing with an online course. But in my four years of programming in R, I've developed web applications, built packages, and analyzed large datasets, all of which will make a strong Python programmer quickly."

We mentioned impostor syndrome earlier, but you really do have to be careful of it when preparing for interviews. When you are looking for your first job out of college or transition to a new career in data science, it's easy to say, "I haven't learned that yet." (Or, at least, easy if you convince yourself it's easy.) Once you've established yourself a little, it can feel embarrassing to not know something. If you don't know something in an interview, don't be afraid to admit it. You can say that you haven't found a chance to use it or that you hope to learn more about that, but it hasn't been a part of your job yet. For example, let's say you're quizzed on machine learning algorithms, but you've been working on statistical modeling, SQL, data cleaning, and working with stakeholders because the data is on such a scale machine learning was done by dedicated machine learning engineers. No one knows everything, and hopefully you've been doing a pretty good job as a data scientist so far—have faith in that. You can show the work you have done and, if you've studied a particular topic before, you'll get it back. It's still always better to demonstrate a willingness to learn rather than trying to fake your way through.

15.3 Finding a new job while employed

If your path to becoming a data scientist included doing a bootcamp, you were probably doing your job search while unemployed. If you were at school, it was expected you might need to take time off for interviews and that you'll be spending time preparing your resume or cover letter. If you're employed full-time though, generally your manager does not want to hear you need time off because you're looking for a new job. So how do you make the time to do it?

For things that can be done any time, like updating your resume and cover letter, researching jobs, sending in applications, and take-home assignments, you should do it on your own time. You don't want people seeing you're doing it and you owe it to your current company to continue to do your job well. But interviews almost always happen during your normal working hours. For a phone call, we recommend just taking it at your work in a phone booth, meeting room, or somewhere else you won't be overheard.

Later-round interviews, however, will generally need to be onsite. If they're just an hour or two and near your office, you can claim a doctor's appointment. If it's longer, and your company allows you to work from home, you can do that and only work part of the day (or try to work a little more later on), but you need to make you won't be expected to take a call or respond quickly when you're in an interview. You also can try to schedule it toward the end of the day and work a half day in the morning.

Scheduling interviews is certainly easier to do if you're looking within your city. If you're looking to move, most companies will fly you in for an in-person day of interviewing on a weekday. In this situation, it's hard to avoid taking the full day off, and most people will usually call in sick that day. But as you can imagine, this is hard to do if you have lots of final round interviews.

That's one of the reasons we recommend applying to jobs strategically. If you have a dozen phone interviews and two on-sites in a week, it's hard to take that time without people noticing, and your job performance will almost certainly be negatively affected. You should be selective in two stages: where you apply in the first place and where you advance after the initial phone call. If you're at a start-up and you want to work at a larger company, don't apply to start-ups, even if the job description looks great. If you're in the initial phone screen and you find out the position is more data engineering and you want to do analysis work, it's okay to stop the process even if they want to continue.

While you can use interviews for positions you wouldn't take as practice, you don't want to do too many of them. Experienced data scientists are highly sought after, which means that you may get a bunch of interest from recruiters and managers once you announce that you are looking for a new job. This is a great feeling—people like you! While it's fine to enjoy that feeling, don't allow yourself to spend time in the interview process with a company that you know isn't a good fit. It's not a good use of your time, even though it is flattering.

15.3.1 Handling your current work

When you're job hunting, it's easy to let your current work slip. To feel okay about moving on to the next thing, you're often thinking about what you don't like about your job, which can sap motivation. But try to continue to do good work: you may need a recommendation from your manager one day, and they are still paying you.

It's possible that during your job search you realize "the grass is not always greener on the other side." In other words, you're not actually finding anything you'd rather do than your current job, or everything is paying much less and offering worse benefits, or they don't give you the flexibility you currently enjoy. It's okay to decide to call off your job search! It wasn't a waste of time if it gave you a newfound appreciation for your current job. If you decide to stay, we recommend returning to our advice from the first section and making sure you've tried to solve any problems you can at your current job.

Going to graduate school

You may decide after working in data science that you want to go back to school to get more formal academic training, either while continuing to work full-time and doing school on evenings and weekends or switching to a degree full-time. If this is something you're thinking about, we recommend you go back to read Chapter 3 where we discuss how to find a good program. We do want to caution to think carefully about if the investment of time and money is worth it, given that you've already proven you can get a data science job. Some reasons going back to school might make sense is if you've decided you want to do a very research-heavy position that would require a PhD, you've gotten explicit feedback from companies you want to work for that you need a master's (not just that you've seen it listed on the job description), or you've found your progress hampered by a lack of certain skills (like in-depth knowledge of algorithms) and free, online options aren't working for you.

If you decide you want to go back to school full-time, you can usually be more open with your manager about it than leaving for another job. If you work at a larger firm, your company might even pay for part of your degree if you do it part-time while continuing to work full-time or if you agree to come back after a full-time degree. Even if not, your manager could be a great letter of recommendation. A good manager knows that school offers you something completely different from a job and should be supportive of your choices.

15.4 Giving notice

If you've decided to leave your job and accepted an offer, you'll need to let your manager know. You should generally give at least two weeks' notice unless the situation is dire. Though unlikely, it's possible that as soon as you give your notice, they will tell you your last day is today. You should be prepared for that and make sure you've sent anything personal you need that's only on your work computer to yourself.

Your boss should be the first person to know that you've leaving. Schedule a meeting with them (call it "career discussion", not "my two weeks' notice") or use your weekly one-on-one time. You want to give notice in-person, if you're co-located with them, or over the phone or video call; don't let them know by email. Start the conversation by expressing gratitude for how they've helped you and the opportunities you had at the company. Assure them you'll do

all you can to help the transition; you can list a few ideas you have, like commenting your code or suggesting someone to take over a share of your work, but figuring that out will be a collaboration with your manager. It's normal to feel anxious about telling your manager that you're leaving, but remember that changing jobs is a normal part of a career.

15.4.1 Considering a counter-offer

It's possible they'll try to convince you to stay by giving you a counteroffer, as it's expensive and risky to hire a new person. They may ask you to meet with your skip-level manager, who has the authority to give you a raise, extra stock options, a one-time bonus, an accelerated review, or other incentive to stay.

There are mixed opinions about whether you should ever take a counter-offer from your current company. On the one hand, they now know you're a "flight risk" and may be reluctant to give you serious responsibilities. It might also cause a strain in your relationship with your manager. On the other hand, they may be willing to solve the main reason you're leaving. It could be monetary, but it could also be switching you to work with a different team. We hope that we have impressed upon you the importance of having open communication with your managers. If you've been open with your manager and still felt the need to move to a new job, it's unlikely that the change you desire can be brought about in a counter-offer. Although we don't love the idea of moving to a new job as a "last resort" and feel it should come well before that point, we have stressed how important it is to communicate your desires before dissatisfaction grows. If you've done that, then know that last minute changes aren't likely to change the overall job environment.

Your manager may try to stress how valuable you are to the team and how difficult it will be if you leave. This can make you feel guilty, especially if your boss or team has generally been good to you. But remember you are not betraying them by leaving. At the end of the day, a job is a job, and despite the rhetoric of some start-ups, a company is not your family. While you should always be respectful and do your best at your job, you don't owe them to work there indefinitely. And you are just leaving a company, you're not dying! If you've grown close to your coworkers, you can still them socially and maybe even work together again one day.

15.4.2 Telling your team

Talk to your manager about how they would like to let the rest of the team know. They may ask that you wait a few days so they can figure out a transition plan and be able to share that with the team when you tell them you're leaving. They might ask if you'd prefer to tell everyone at once in a regular team meeting or if you want to meet with people one-on-one. We recommend considering the size of your team when thinking about this. If you've worked with the same five people consistently over years, you may want to tell them individually. On the other hand, if you're on a 20-person data scientist team and worked with a dozen stakeholders, it would be emotionally exhausting to meet half an hour with all of them.

One mistake to avoid here is scheduling meetings with people before talking to your manager, even if you schedule those meeting for after your manager conversation. If your coworkers become suspicious of why you suddenly want to meet with them and ask if it's because you're leaving, it will be really awkward: you'll either have to lie or tell them before your manager knows.

You will be asked by most people why you're leaving. Make sure you have something to say and try to keep it positive, focusing on the new opportunity and what you're grateful for at your current company. Even if you've become friends with a coworker, be wary of being negative just the same. Remember that you may desire to come back at some point, and you don't want a reputation as someone who trashed the place on your way out. Some people have very close relationships with their managers, but even if that's the case, you don't want to talk too much about the negative aspects of your current job as this can damage your friendship and is also usually unnecessary. Among the other good reasons for exiting gracefully is that keeping a good relationship with your past coworkers and managers can be invaluable career-wise as you may run into them again or need recommendations later on.

As you say your goodbyes, send people a way to contact you after you leave (email, LinkedIn, twitter). It's nice for coworkers to be able to stay in touch and it's also a good way to ensure that you are part of a functional network for yourself and for others.

Checklist for before you leave

Before you head out, there's a couple of administrative things you'll want to make sure you have:
The contact information for HR, in case you need something later like information on your stock options
Any personal pictures, passwords, or files you only have on the company computer.
Benefits and stock compensation portal log-in information.
Copies of employee agreements, offer letters, and termination agreements.
Information on how your vacation will be paid out if you have any left.
If you're not starting a new job immediately, what your health insurance options are to continue coverage.
If you contributed to a flexible savings account for health or dependent care, what the last day you can spend it is (generally either your last day or the last day of your last month of employment). These are "use it or lose it" funds, so if you don't spend them they'll be gone.

15.4.3 Making the transition easier

The best way to leave on a good note is by making the transition as easy possible. You may not be able to find a replacement, but you can set the team up for success while they're backfilling the role (if they chose to). Make a transition document for your manager with your responsibilities, which ones you can wrap up, which ones need to be transferred (and suggestions for who could pick it up), and which ones will have to wait until there's someone new in the role. Beyond informing the person taking over that project, you might need to make introductions to outside partners or clients or let them know that you will not be handling the project any longer.

Try to clean up any loose ends. If you have work that may be useful for others but is only on your computer, add it to a git repository or share the Google doc with someone else. You'll probably be given little work in those last few weeks since people know you won't be around long, so it gives you time to do things maybe you were too bogged down to do before like documenting all of the processes you've created. Some things you could do include:

- Adding tutorials: Have you been the "go to" person for a certain topic, like how the finance data is organized or best practices for A/B testing? There's no way to replace you being there, but by making presentations, internal posts, or documentation, you can help fill some of the gap you'll leave.
- Organizing your files: Even if you add everything to GitHub, it's not going to help anyone if it's 100 files with names like "random_stuff" and "misc_analyses." While you may end up needing to dump a few files in an "extras" folder, try to make the files easy to navigate and add explanations where necessary
- Adding comments and explanations to analyses: ideally, for any impactful analyses, you'd already written up the findings, linking to the code which included comments. If there are some you didn't have time to finish though and you think it would be valuable for someone to continue the work, you can flesh them out. While you don't need to comment every piece of code, it can be helpful to explain some surprises in the data (and how you worked around it), what you've already tried, and explanations for why you choose analytical methods.

The worst thing you can do is forget that you are the only person who knows how to do "x" at the company and leave that unassigned. If you do, you may get frantic calls/emails about how to do it while you are trying to adjust to your new job. Forgetting that you own the only password to a particular system is a good way to get in trouble even after leaving. Some employers don't know how to say "goodbye and good luck" and may call about projects you were working on. For your sanity, it's best to be able to refer them back to your exit documentation until they get the hint that you no longer work there. Even if they don't follow up, you won't find much value in the network you built there if you left a huge mess for them to deal with.

We hope this chapter has made it clear that while the uncertainty around leaving your job may be stressful, it's a normal process and there are ways you can approach it to make it smoother. As we've discussed many times in this book, few decisions you make are final: just because you start looking for jobs doesn't mean you have to leave, and even when you leave a company you may end up coming back in a few years. The most important thing throughout this process is to make sure you're focusing on how to make your job best fit your career goals.

15.5 Interview with Amanda Casari, Engineering Manager at Google

Amanda Casari is an Engineering Manager at Google on the Google Cloud Developers Relations team. Previously, she was a Principal Product Manager and Data Scientist at SAP Concur. She also served five years in the Navy and holds a Master's in Electrical Engineering.

15.5.1 How do you know it's time to start looking for a new job?

My advice for people is to understand what kind of work they want to be doing and if that fits with the role and where the product, team, and company is at. For me, I do very well in times of high change. I enjoy working on projects at their start during the ideation phase, but I also enjoy sunsetting products. A data science job where I was spending the majority of my time optimizing models for single digit percentage increases or doing hyperparameter tuning, on the other hand, wouldn't work very well. I also think about the stage of the team cohesion. Do you want to join a team that has a strong bond and culture already, or do you want to join one that's just forming? Overall, my role, where the product is in this product lifecycle, and where the team is in forming vs. conforming to a culture, all influence whether or not a role is still a good fit for me or if I should look for something more challenging.

15.5.2 Have you ever started a job search and decided to stay instead?

All the time. When I'm looking at other roles I may identify things I could actually be doing at my current company. It's kind of a kick to go out and find those opportunities as opposed to waiting for somebody to give them to me. This fits into my broader philosophy that your current job responsibilities should be a conversation you're having with your manager. For the engineers I manage, I try to have open and honest conversations where they let me know what opportunities they're looking for. I can then figure out whether that exists within the current team or, if not, whether we can find them a 20% project outside of our team. This way, they can explore it to see whether or not that's actually what they want to be doing.

15.5.3 Do you see people staying in the same job for too long?

Oh, yeah. I've seen some people get a kind of hero complex, where they feel like nobody else could possibly do their job. And the real answer is nobody else is going to execute your job exactly like you do, but that doesn't mean that no one else can do it. Sometimes somebody sticking around teams can be very harmful *because* they remember all the problems and small decisions the team has ever made. They may point out, "We tried that idea before two years ago and it didn't work, so you shouldn't possibly try it." But that can hold the team back by not focusing on what's currently possible, just decisions made in the past.

I've also seen folks who have become fairly jaded about management, where they spend a lot of their time just complaining. They'll always have an open door when there's gossip for how things are going, and that's really negative for a company. You don't want people who are

going to be taking their discontent and turn it into something that is infects the rest of the team.

Finally, I've seen folks who are not being challenged by their current roles and are simply doing what's asked of them and no more. This can be fine when you're newer, but for experienced people and those in leadership, I expect something more than that. I want to see experienced people having larger kinds of impact and change. If you see a problem, you should think about addressing it in a way that scales, is repeatable, and solves an organizational problem, not just something that is a one-off fix.

15.5.4 Can you change jobs too quickly?

When I'm looking at job applicants, I might question if somebody had had a job for less than a year. As a hiring manager, what you're trying to understand by looking at people's job tenures is if they're going to leave in a few months, because hiring and onboarding someone is a long and expensive process. But while in other industries you may see 2 to 3 years as the "minimum," in technology I see that as a long time. 2 to 3 years is multiple projects with multiple cycles in technology, so leaving by then makes sense to me—really anything more than a year does. If you have to leave earlier than a year for your mental or emotional health that makes sense too; it's not worth any job to jeopardize those.

15.5.5 What's your final piece of advice for aspiring and new data scientists?

Find your community and people who can help you. I have benefited so much from having a dear friend who's sponsored me by recommending me for speaking opportunities and talking me through job offers. Having a person with experience to talk through those details is invaluable and has really helped me understand my worth and feel confident walking into a new position. For finding community, there are so many places that you can belong, but there may be some where you don't feel that sense of belonging. You don't have to stay in a place where you feel uncomfortable, whether it's because of the language, the people who are gathered there, or the focus of the group. And if you can't find a space you feel comfortable, look for someone who you would want to be in that kind of group and ask them if they can help you form one.

15.6 Summary

- When deciding if you should look for a new job, four questions to ask yourself are: are you still learning, have you talked to my manager and seen if my responsibilities could change, are your manager's career goals aligned with yours, and have you thought about what you're looking for (and not looking for) in my next role?
- While many principles of having a successful first data science job search (which you can find in Part 2 of this book) still apply, for your second job you should also reflect on what you've liked (and not liked) in your first, prepare to share your experience in a positive way that respects confidentiality, and plan for how to juggle interviews with

your full-time role.

- After you've given two weeks' notice to your manager, focus on how to make the transition as easy as possible for your teammates by tying up any loose ends, documenting anything that currently only lives in your head, and sharing any helpful code.

16

Moving up the ladder

This chapter covers

- **Different paths beyond senior data scientist**
- **The opportunities and risks of possible career trajectories**

Through the last part of this book we've covered how to flesh out your career—learning to cope with failure, joining the community, and switching jobs. As your career strengthens you'll eventually want to decide where that career is going to go. It's not obvious what are all the options for data scientists as they progress upwards—becoming a manager is one possibility, but it's not the only one. In this chapter we'll cover three common career trajectories for a data scientist—moving into management, becoming a technical leader, switching to independent consulting—and the benefits and drawbacks of each.

The management option is what most people think of when they think about career growth—managers are the people who lead teams, including deciding hiring and promoting, setting strategy, and giving career mentoring. Principal data scientists are people who are masters of their field, and companies rely on them to solve difficult technical problems. Independent consultants are data scientists who have enough skills and a network that they can freelance for a living. See Figure 16.1 for an extremely high-level summary of the paths in this chapter.

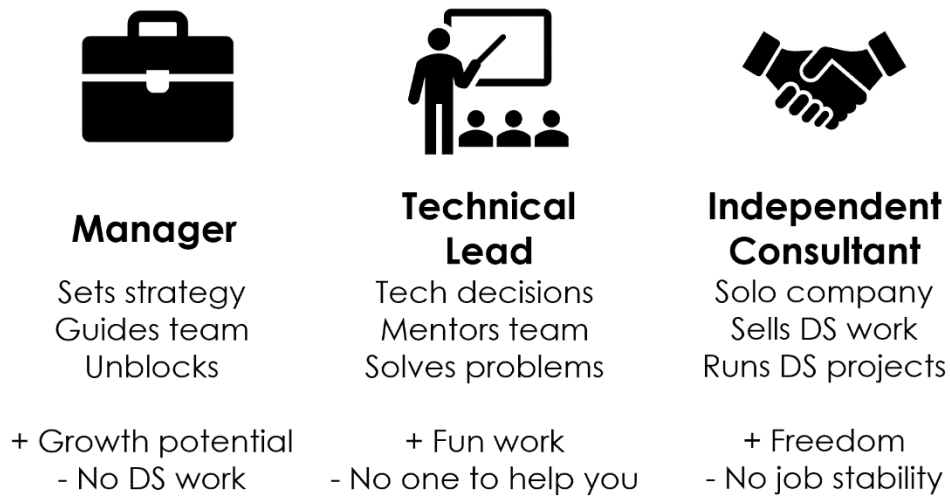


Figure 16.1: the paths in this chapter

As you continue to build your career you will want to focus on taking one of these paths. By having a clear goal, you are more likely to achieve want you want. That being said, the closer you get to achieving an opportunity you want, the more you may realize it isn't something you want after all. Fortunately, making changes like becoming a manager then deciding you don't like it, or going from industry to consulting then back to industry, is not unusual—and while sometimes reversing decisions you've made can be difficult it's frequently done. Learning from your missteps is a fast way to grow as a person!

Knowing what level you are

As you grow as a data scientist, you'll be gaining valuable skills (just like the last 15 chapters have covered). But as your skillset and professional maturity grow, at some point you'll no longer be working like a junior data scientist. It's difficult to know exactly when that transition happens, which makes deciding the right time to advocate for a promotion or new job title difficult. Keep in mind that each company will have their own levels and expectations, so the same title may mean very different things are two different companies. Within a company there may even be a skills matrix describing exactly how the levels differ, but people may interpret the matrix differently and there is always ambiguity. To help you, here is our high-level guide to the different expectations for levels of data scientist.

a. Junior data scientist – a person who can complete a data science task given clear direction on what that should be. For example, if told to use a clustering algorithm to segment new customers by their purchasing attributes, a junior data scientist should be able to do this with guidance from a manager. If there are technical problems occurring like bugs in code or data systems not connecting, they may need to consult with other team members to solve.

b. Senior data scientist – a person who can not only complete a data science task, but can figure out what other tasks are needed. Not only would they do the clustering implementation from above, but they would also realize things like the same algorithm could be applied to existing customers (and then they would do it). Not only are they skilled at solving technical issues themselves, they also are the people brought in when others have problems.

c. Above senior data scientist – At levels higher than senior data scientist, the role becomes more about helping others (which will be covered more in this chapter). Thus, a person usually moves beyond senior data scientist when they are consistently mentoring others, creating strategies, and seeing the bigger picture.

16.1 The management track

"Going into management" can seem like the default option for data scientists as they go farther into their career. It makes sense from exposure—everyone has (or has had) a person they report to. Despite this, the day to day tasks of a manager can often still be mysterious.

A manager is a person who is responsible for a team of people successfully performing whatever their objective is. These five basic tasks are usually, but not always, the job of a data science manager:

- **Deciding the team's work.** This can be at a strategic level like deciding what large projects should be taken on. It can also be at a more tactical level like making the calls on what features should be included in a product.
- **Determining who should be on the team.** A manager usually decides who to hire and who to let go, with the approval of HR and other parties. They coordinate the interview process and weigh in during it.
- **Mentoring team members.** Everyone on a team has their own unique challenges to work through, and a manager helps them do it. A manager regularly checks in on each person and provides advice and recommendation to help them solve their problems.
- **Resolving team issues.** If the team is having issues that prevent work from being done, such as another team in the company being unwilling to provide necessary access to data, it's the job of the manager to find a solution so the team can continue.
- **Project management.** A manager has to keep track of the work going on within the team and ensure things are on schedule. While many teams have a project manager specifically assigned to this task, the manager still needs to keep oversight on the work.

Together, these tasks cover a wide range of work and a high amount of responsibility. A manager has to constantly be communicating with people inside their team and outside of it. They need to be aware of how team members are doing, how projects are doing, and what is on the horizon. It's an immense amount of work, and if any of these tasks are not done well the whole team will suffer for it.

Importantly, none of the basic tasks of a manager are technical—a manager generally isn't machine learning models or providing analyses that help the company make decisions. A

manager doesn't have time to do that sort of work—and even if they did it would be better for a data scientist reporting to the manager to do it. That means becoming a manager involves giving up much of what drives people to become data scientists in the first place—a desire to use data to solve interesting problems. Instead, their job becomes to support other people doing that.

16.1.1 Benefits of being a manager

Being a manager has a bunch of perks. First, if you're a person who hates when other people do work that you think is silly, then being a manager means you get to influence that. For example, if you think a particular machine learning model would be helpful for the company, you can assign the team to make one, and if you think it's not a good idea then you could have the team avoid it. It's incredibly fulfilling to get to choose the direction of the team and then have the team succeed. While you don't have total control, as sometimes people higher up will demand something is done or people on your team will strongly recommend doing (or not doing) something, you have a large say.

Moving up into management will often give you an initial pay raise from being an independent contributor. Management also opens the door to further roles: senior manager, director, or vice president. Each of these roles comes with higher pay and broader leadership across the company. You may even get to a level where you oversee more fields than just data science, such as customer research or software development. You can even transition out of data science entirely and only lead other areas.

If you like teaching and helping others, management is a great role for it. Much of your job is helping people on your team succeed by working with them, teaching them what you've learned in your career, and helping them with their struggles. A great manager is like a business therapist—they sit down for 60 minutes with a person and help them understand their problems and work through them.

Lastly, you can have an immense sphere of influence as a manager. Being the person who makes the final decisions on whether to build a new product or expand to a new country is really fun! As you get higher in the company, you'll be able to influence more and more. If you follow this path, you have the possibility of some day running a company.

16.1.2 Drawbacks of being a manager

The biggest downside of management is that you will stop having time for doing data science. Your job will be filled with things like talking about data science, mentoring data scientists, and thinking about data science strategies, but you won't be doing data science yourself. Your day may be filled entirely with 30-minute meetings ranging from deciding team strategy, to buttering-up stakeholders to get funding, to one-on-one meetings to help under-performing employees learn how to improve.

No longer doing data science as your job could be negative for two reasons. First, if you became a data scientist it's probably because you liked working with data—so you are giving

up the job you've trained so long to have. Second, by not doing data science, over time you will become out of practice and disconnected from the latest changes. This means that if you decide you don't like management and want to go back to being an individual contributor, you will no longer have the skills to do so.

Another downside is that you are still limited by the management above you. You may have great ideas for your team's strategy that get shot down by your boss. Then you have to lead your team down the path your boss set even though you don't agree with it. Having to keep a positive attitude about work you don't agree with for the benefit of the people on your team can be extremely frustrating. If you let your frustrations be felt by the people under you, that can lower their job satisfaction.

Being a manager gives you a lot more things to worry about. You have to worry about your own performance just like when you were an individual contributor, but you also have to worry about the performance of the rest of your team. You have to worry about their career satisfaction, and you have to worry about what's happening politically at levels above you. You have to worry about if the team will have funding and if a project going too slowly will get the whole thing canceled. Having this many worries, most of them not directly in your control, can be extremely stressful for people with certain personality types. If you are a person who has trouble not taking their work home, management may not be for you.

Lastly, managing people is a totally different set of skills from being a data scientist. When you switch to being a manager you'll have to learn these new skills and return to being a beginner at your job. Switching from being great at your job to being a novice can be stressful and miserable. While you'll eventually get the hang of it, the journey to becoming a good manager is a long one.

16.1.3 How to become a manager

If you are an independent contributor and you want to be a manager, you need to find opportunities to develop and practice leadership skills. These include working well with others, both more junior and senior than you, seeing the bigger picture, and managing a timeline for a project.

Unfortunately, there isn't a single course you can take to learn these skills--the best you can do is find situations within your current work where you can grow. This might a small initiative within your team that someone needs to take charge of, like setting up a new software stack or coordinating the rollout of a new model. The important component of these situations is that you are the one leading them and making the decisions. This may feel extremely unnatural at first, and that's totally normal and will fade. Books on management and business can be somewhat helpful, but only if you have opportunities to use what you learn.

Once you feel that you have grown the skills be a manager, the next step is to find a role.

A PROMOTION FROM WITHIN THE COMPANY

Often the most straightforward way become a manager is to get promoted within your company. This can be the easiest way because the people deciding to put you in the role are the ones who have watched your skills grow in your current job. The difficulty with this route is that the company has to have a need for a new manager--either your current manager has quit or been promoted, or a manager role needs to open in a related team. Depending on your company this may rarely, if ever, happen.

GROWING A NEW TEAM YOURSELF

Another route is to become a manager by growing a team yourself. This can either be by starting a project in your current company that ends up requiring more people, or by becoming the first data scientist at a new company and then growing a team there. This route can be extremely fulfilling, since by growing the team yourself you have a strong hand it who is on it and how it functions. It requires you to be in the right place, at the right time, and to be a strong enough leader that you can quickly grow the whole team infrastructure. Sometimes the people who do this are called "player-coaches" since they are both functioning as the team's first independent contributor and becoming a coach for others. These opportunities where you can create your own team from scratch are exceedingly rare.

GETTING A MANAGER ROLE AT A NEW COMPANY

The last approach is to get hired into an open manager position at a different company. This route relies on you being able to show that you have management skills without having technically been a manager before. On your cover letter and when interviewing you'll need to talk about all the projects you've been leading and people you've been mentoring while an independent contributor. You should be able to tell by the amount of attention your resume is getting if you have a chance with this route--companies are desperate for good data science managers, so if your resume looks good you should be getting responses.

Learning to manage – Rob Stamm, Director of the AI @ T-Mobile team

I learned a lot about being a manager during my first experience in a full-time manager role. Before that I had been a product manager, which meant guiding the direction and development of a product (but not people). Then as a newly-hired senior manager, I had to oversee several different product managers and help them do their jobs. I ended up failing miserably in this role. I wanted to be a senior manager and lead a team, but I wouldn't stop doing the product management role too. I wasn't letting the team do their jobs–I kept doing it for them.

Four months into the job, one of the product managers walked into the office and told me they were going to quit, and my behavior was why. That was a huge realization for me that you can't both be a manager and continue to try and be an individual contributor. It was the first time someone pointed out that a manager has to enable their team to make decisions, and that stuck with me.

Eventually I learned and grew from that experience and started leading bigger teams and larger projects. Being a leader is incredibly rewarding: I have the job of helping my team be their best, and I feel gratified when I see the results of that. One particular project at T-Mobile that was especially fulfilling was getting to help the AI team start from just an

idea and a few dollars into funding for a large-scale team. While I wasn't designing the product or writing the code, I got to help the team when they were stuck or needed resources like money or people. That's rewarding in its own way.

16.2 Principal data scientist track

A principal data scientist (or staff, chief, data scientist V, technical lead, or other title depending on the company) is a person in an organization whose job is to be an expert in data science and help others with technical tasks. Becoming a manager involves steadily doing less and less data science, while being a principal data scientist is to do more and more of it. But instead of just doing data science on your own (although you'll do plenty of that), you'll also be tasked with helping other data scientists with their jobs. A principal data scientist is a technical leader that others will rely on.

Principal data scientists are often paid as well or better than managers. They are people who start as junior data scientists then get promoted to senior data scientist and continue to grow beyond that. As they grow in their data science careers, they get more experienced and more mature in their ability to understand problems and how to solve them. Soon they know so much that when other people are struggling, they are able to go in and quickly assist. People throughout the company come up to them asking for suggestions on how to handle problems and what is likely to work or not.

A principal data scientist job involves multiple responsibilities:

- **Influence data science strategy.** A principal data scientist has to layout the plan for how to tackle data science problems. Is modeling payment fraud feasible? Should a neural network be used? The manager is responsible for the idea and business plan; the principal is on the hook for how it should work.
- **Mentor junior data scientists.** Since a principal data scientist has so much experience, it's their obligation to share what they know with more junior employees. Their growth is as important as the principal data scientists' own work.
- **Find solutions to difficult problems.** When the data science team is plagued with a difficult technical problem, the principal data scientist is the person on the hook to devise a solution—or declare it impossible to solve.

Compared to a manager, a principal is still very much doing data science work. Thus, this is a great role for people who love being a data scientist. If all you can think about is data science and you love going to conferences, being part of the community, and learning more about techniques and methods, this can be a great role. Because of how critical a principal data scientist can be for a team, it requires a person with maturity, responsibility, and experienced enough at data science that they can quickly ramp up in new areas as needed.

Asking for a promotion

At some point you'll likely find yourself in the position where you think you're ready to be at a higher level and yet you haven't been promoted. This can be frustrating situation (you can see why you're ready, why can't everyone else!?), but it's not a hopeless one. The best thing that you can do to help move the process along is advocate for yourself. Let your manager know that you're interested in moving to a higher-level position, and that you want to work with your manager to create a plan to get there. If your manager is good they should be happy with this—by being clear that you are ready to make changes to achieve the next position, you're opening the conversation as to how that can happen. Try and set a goal that has a specific date, such as the next performance review cycle, to get promoted. The goals should be specific as possible, like "perform three technical presentations in the company" or "create and deploy an entire machine learning API on my own." By having clear goals with a timeline you'll be able to have ongoing discussions on if you're making progress.

If your manager is giving you feedback as to why you aren't ready, listen to it. While it's sometimes hard to hear negative feedback, your manager has a perspective that you don't on what you need for the promotion. If your manager is telling you that they think you are ready and want to advocate for you in the promotion process, try and give them as much documentation about what you've been doing and why you're capable for the new job as you can. That will be fuel to help your manager move you on.

If it seems like no matter what you do you still can't get the promotion you seek, that might be a sign it's time to switch to a different company. There are plenty of occasions where people are so used to seeing someone in a particular role that they are unwilling to take a risk on having that person in a new one. By changing companies you're getting to work with a group of people who don't have expectations about what you can do and may give you more opportunities.

Although the work of a principal data scientist is related to data science, it's rare that a principal would be doing individual-contributor work like making a single analysis or creating a machine learning model. Those tasks require a lot of time and unified focus, and a principal data scientist has to split their work across many projects and areas. Data science projects tend to fall on junior and senior data scientists, while the principal coordinates and oversees them.

16.2.1 Benefits of being a principal data scientist

A principal data scientist often gets the most interesting problems. If the team has an idea for a totally new data science approach no one has tried before, you're going to be involved in trying it for the first time. If there is a complex technical stack that needs to be integrated, you're going to be there. If a project is supposed to be launching but the team just can't get the model to work, you're going to be there. Getting to be in the middle of the action can be enormously fulfilling. It's interesting on the technical side and you get to feel validation as you prove yourself useful over and over. The team will be aware that simple, menial data science projects aren't worth your limited time, which means you'll have to do less than most data scientists.

Your manager will also understand just how important it is for you to keep up with technology, which means you'll likely have funding for going to conferences and the time to toy around with new technology. Depending on your manager, you may be expected to

present at conferences to promote your company, but since you've been working on interesting problems, you'll likely have work worth sharing. If you ask your manager for resources, like money to try out a new cloud service, you'll usually get it. Your manager will trust you to use your time and budget effectively and not be wasteful with it, which is not something every data scientist gets.

You'll also get to talk about data science all the time. Since you'll be mentoring junior data scientists, you'll get to tell them about different approaches, work with them to hone their ideas, and point out areas where data science approaches could have problems. For a person who likes data science this can be quite fun.

Making the data science plans can be very empowering. By getting to be the person who decides what types of models to use, how to structure data, and how to scope a project, it's more likely that projects are done the way you want them. And since you're an expert in data science, the projects should be more likely to succeed! More junior data scientists don't always get to devise a data science approach, execute it, and see the results themselves without having someone else make key decisions in it.

16.2.2 Drawbacks of being a principal data scientist

The biggest problem with being a principal data scientist is that you won't have someone to turn to for help with you are stuck. A junior data scientist usually has a mentor or more senior data scientist they can ask questions to, or they can even do Google searches to find answers to their problems. As a principal data scientist there probably won't be data scientists more senior than you in your team. The problems you will be faced with will often be so unusual or unique that no one has faced them before—and so no Google search will provide an answer. Thus, you'll need to be able to work in environments where you don't have help. This can be crushing for many data scientists.

While you'll get to be involved in the most interesting data science problems, you may also be faced with the most annoying ones. For example, if a dataset is stored as terabytes of poorly-formatted csv files on a server no one has touched in years and with a schema no one knows, you'll be called in to figure out how to use the data. That's not really an interesting problem, it's just a minefield that no one else can navigate. There are many problems like this you'll face, and you probably won't be able to delegate them to other people.

Because your knowledge is so vast, you'll be in high demand—and from that you'll be very busy. You'll frequently find yourself with more work than time to do it, and you'll have to let interesting projects go just because you don't have the bandwidth for them. It's easy to fall into the habit of working more hours than you should and still not feeling like you've done enough. Since so many people rely on you, having only a 40-hour work week or weeklong vacations without your laptop are a lot harder to pull off. If you want a nice relaxing job, a principal data scientist probably isn't it.

16.2.3 How to become a principal data scientist

If you work as a data scientist and continue to grow your career, you "default" into becoming a principal data scientist. The role is the natural progression from being a senior data scientist; it's an expanded version of the role. Unfortunately, there are many senior data scientists who struggle to get promoted to the next level. To be qualified to be a principal data scientist, you need to be strong enough of a data scientist that you can effectively work independently and can lead others. You then need to draw attention to your abilities and contributions and find other people in the company to champion you so that you are well-known as a critical part of the team. With that you can advocate for promotion to this higher role.

To work independently as a data scientist, you need to be able to handle full projects without outside guidance. If your manager gives you a task like "create an analysis on where we should place our next retail store," they need to be able to trust that you can do that without the help of others or getting stuck and not telling anyone. As you grow as a data scientist, the ability to work independently should come somewhat naturally as you become more practiced at the work. To speed up this growth, try to pay attention to when you get stuck and what you do in those situations. If you ask for help—what does that provide that you couldn't do on your own? The more you can maximize the times when you can solve the problem yourself, the better.

As you gain experience, you should also pay attention to the data scientists around you—what are their problems, and are you able to help them? If you're a more senior data scientist, it's likely that junior employees are dealing with issues you've faced and solved in the past. The more opportunities you can find for technical mentorship, the stronger you'll get at helping others—which is great for a principal role.

Lastly, as new ideas are thrown out, try and look for situations where you can create the approach. For example, if the company wants to find the location of retail stores, you could come up with the idea of using location optimization techniques. Your ideas may or may not work, but if they work it'll look great and if they don't, you'll learn more about how to come up with ideas. It's easy to rely on other data scientists to formulate the approach, but it's hard to be a principal data scientist without this skill.

16.3 Switching to independent consulting

It's a common dream to be your own boss and have your own company. In the case of data science, that usually means being an independent consultant—having a business where companies hire you to work on specialized data science projects. In theory, companies will only want to hire an outside consultant if they need a special set of skills for an important problem. And if you're running your own company, you get to keep all the revenue for yourself, so no money is lost on the fat-cat executives above you. People will hire you because they believe you're a great data scientist, so you'll be valued for your expertise.

As an independent consultant, you have to be an entire company at once. That means you have to be doing many different things, such as:

- **Marketing your business.** You won't be able to get new clients to your company unless people know about you. That may mean going to conferences, having meetings with old colleagues, or creating promotional materials like blog posts.
- **Doing sales.** Once you find a company interested in hiring your company for a data science project, you'll need to meet with them and make a proposal for the work. If you can't get them interested with this, you won't get work.
- **Executing the project.** This is the data science work you were hired for. It also includes the project management to keep things on track and deal with any situations that may arise (like bad data).
- **Delivering the results.** Once you've created the model or done the analysis, you need to present what you've done for the client and get them on board with it. If they like it then you may get a follow-on project, and if they don't you can lose them as a client.
- **Managing the business.** Businesses need to do things like pay taxes, create legal documents, keep track of accounts and cash flow, and lots of other small tasks that add up over time.

You need all of these skills, which span far beyond that of a data scientist, to be effective as a consultant.

Depending on the type of clients you get as a consultant, your job is likely to be half data science, half all the other work to keep the business going. There will be a certain flow with this, where you'll be working on one project for a client company, and as it nears the end you'll be close to sealing the deal on another project for a different company. The work will come in at a chaotic rate: you might find that in one month you have three companies all asking for your time and the next month no one has work available.

Finding clients is often the hardest part of being an independent consultant, and it requires dedication and a strong network. Most clients tend to come either directly from or thanks to recommendations by former colleagues or clients. The more people that know the consultant and can vouch for their work, the more work will come in. Thus, to be successful many, many people, ideally those with the authority to hire a consultant, have to know about the consultant's expertise. The more diverse the network of connections is across different companies and industries, the more likely it is that the work will come in at different times. Having such a strong network requires that the consultant have worked with many different companies before—either by having changed jobs a lot earlier in their career or by having been a consultant at a larger firm.

If you're successful as an independent consultant, you have the opportunity to start hiring more people and grow the business. The company can go from just you, to a team of five, to an organization of one hundred. As the CEO and founder of the organization, you'll get to lead it in the direction you want with the culture you want. By taking a cut of the money all the other consultants bring in, you'll find yourself wealthy. While this is a rare outcome, this outcome is by far the most profitable in this chapter of any of the paths.

16.3.1 Benefits of independent consulting

As an independent consultant, you get to be your own boss, which means you get to choose whether to accept possible data science projects, what approach to take, and how to present the results. You don't have to rely on anyone else, which for some can be incredibly liberating. If you can keep your expenses down, keep your rates high, and keep a consistent set of clients, then your business could potentially be quite lucrative. You have the opportunity to make twice as much or more as you would working for a company. If you want to work from home or take a day off, you have the power to do that without having to argue with someone.

You have ownership of what you make too. If you come up with an interesting method of solving a problem, you can decide to patent it or market it as your company's product. No one has the ability to take away your work, whereas if you work for someone else's company the company can claim your ideas as their intellectual property. If you are able to come up with a portfolio of useful products, then that can sustain you for years.

Consulting can be fun! There is a certain thrill factor to flying around the country, helping people with your ideas, and doing it under your own company's name. It can feel incredibly validating to have many people wanting to pay money for your time. It can also feel great to deliver a solution that the client likes and know you did that yourself.

16.3.2 Drawbacks of independent consulting

The drawbacks of being an independent consultant are staggering. They are so onerous that bold fonts will be used liberally in this section.

Independent consulting is wildly stressful. Whether or not you get paid in a month is entirely dependent on if people decide to hire you, which is often due to factors outside your control (like company budgets). Conversely, you may find yourself with more work than you can handle, and you have to figure out which project you're going to let slide. You often have to sell consulting projects before getting full access to the data to see if the project would even be feasible—and you have to figure out what to do if it isn't. There are a thousand ways that being a consultant can keep you up at night.

Independent consulting can make you broke. If you become a full-time independent consultant and can't find opportunities, you will lose money—fast. Even if you can sign a contract to do consulting, large companies often won't pay until 90 or 120 days after the work is done, which can be a half a year after you started. If you aren't able to handle those wild swings in cashflow you won't be able to be a consultant.

You really won't have anyone to turn to. If you're working on your own as a consultant, people are bringing you on because they have a problem they can't solve. That means there probably won't be people to bounce ideas off of and you are on your own. If you're having trouble with an analysis or getting a model to work, you'll be forced to find a solution yourself or you'll have to tell the client that you failed.

Your work won't be that much data science. The amount of time you'll spend doing marketing, negotiating sales, writing contracts, and keeping up with accounting will feel

immense compared to the time you take doing data science. Being a strong data scientist is not enough-all of that other work is necessary for your consulting company to survive.

16.3.3 How to become an independent consultant

To become an independent consultant, you need to have a strong set of data science skills and a track record of solving problems independently on the job. You'll also need a network of people who know your abilities—either from working at multiple companies or even better by working at a large consulting firm.

You can test the waters by doing freelance data science work in your free time. Put up a website, post on LinkedIn, and let people know that you're available to help. If you're able to get clients, you'll learn more about consulting as you do freelance work in the evenings. If you find you don't have the energy to do work in the evenings, you probably won't like being a consultant. If you aren't able to get freelance clients then that's a sign that your network isn't big enough, and you should focus on that first.

If you find that you have so much freelance work available that it's difficult to continue with your full-time job, then that's a sign that it's a good time to transition to full-time independent consultant. At this point you can start focusing heavily on consulting and if you can find a core set of clients to start with, you can quit your job and move to full time consulting.

Leaving data science

One last route is to leave data science all together. Maybe you find the work no longer interesting. Maybe the workload doesn't align with your work/life balance needs. Maybe you've found yourself using data science for unethical purposes and you can't bring yourself to do it anymore. There are plenty of reasons why data science isn't the right field for everyone, and there is no shame in that.

It's difficult to give suggestions on how to leave the field altogether since it's so dependent on what your next field is. A data science resume is an easy hire for related fields such as software development or engineering. Other fields may be harder to transition to, but just like in Chapter 6 where we suggested you make previous roles sounds as much like data science as possible, you can also try and take data science roles and make them sound as much like other field as you can.

If you do leave you may find yourself later wanting to return. Once you've left data science, as long as you periodically do a small project or try to keep up with the field, it shouldn't be hard to catch up with what you missed when you're ready to return. In that case it'll be like starting at the beginning of this book again, only with a lot stronger of a background. While data science has lots of marketing and buzz right now as a hot field, don't let that make you feel you have to stay—your happiness is the most important. Do what's right for you.

16.4 Choosing your path

In this chapter we've presented three paths to grow in data science, but there are many others too. All these choices can be overwhelming, and for the most part you can't try a path

before making a heavy commitment to going down it. How can you know which career path is right for you?

The raw truth is that you can't know—you can't know which choice is "right" because there is no right choice. These decisions are so dependent on the companies you're working with, the people around you, and your personal interests at that point in your life. You can only make the choice that feels best for you, and not worry too hard about the missed opportunities.

This is the same lesson that has shown up over and over in this book. Just like there is no one right way to learn data science skills or one right type of company to take a job with, there is no best way to navigate the senior parts of your career. You can only do what is the best for you with the knowledge you have. Hopefully this chapter has provided you with enough knowledge that navigating these career choices is a bit easier.

16.5 Interview with Angela Bassa, Head of Data Science, Data Engineering, and Machine Learning at iRobot

Angela Bassa is a Director at iRobot, where she oversees data engineering, data science, and machine learning work across the organization. She previously worked as a consultant and as a senior manager and director of analytics and holds a bachelor's in Math.

16.5.1 What's the day-to-day life as a manager like?

It really depends on the complexity of the organization, which is often a function of its size. When you have three people, you have three edges that connect them; when you have seven people, you need exponentially more edges to still connect everyone to each other. If I need to coordinate across different products, teams, objectives, and timelines, then it's going to take a lot of meetings. I spend about a third of my day on that strategic coordination to make sure we're working on the right things in the right way for the right reason. Another third is spent working with my team, generally by being a sounding board and helping give them context or feedback. The final third is administrative. For example, is the budget in line? Does everybody have the money for their training and development that they signed up for? If there's a really interesting women's conference coming up and I have a bunch of openings in my team, do I want to sponsor that conference?

16.5.2 What are the signs you should move on from being an independent contributor?

Deciding to become a manager takes a lot of introspection, self-awareness, and being open minded. Finding the point at which becoming a manager is most likely to succeed has a lot to do with being at a place (both professionally and personally) to make such a large transition. Management is a different profession: it has a different skill set and a different risk profile. If you mess up as an individual contributor you're really only in charge of your own destiny. When you're managing other people, the responsibility that you're dealing with is somebody

else's access to healthcare or somebody else's ability to pay for their rent. But I do think anyone can be a manager, and if you're worried about being a manager then you're probably a great candidate.

16.5.3 Do you have to eventually transition out of being an independent contributor?

Data science as a profession is so new that there's still a heavy self-selection of who chooses to go into this profession. A lot of us are ambitious go-getters: we're paving a brand-new career because we're the kind of people who do that with our lives. But if you look at other career paths like accounting, they're not "up and out;" you can absolutely be a senior accountant for a very long time. What can happen is that you may hit a limit to compensation growth or growth in your knowledge. If that is something that meets your career objectives, I see no problem with somebody who likes what they have staying right where they are. Still, there are so many talented professionals flooding data science as a career that if I find somebody who is eager and gunning for it then I am more likely to be biased towards that personality type when deciding who to bring in.

16.5.4 What advice do you have for someone who wants to be a technical lead but isn't quite ready for it?

Find a "sounding board" person that you can have a frank and open conversation with so that you're not just in your head. That can help you get concrete feedback and understand the things that you need to get better at. Having a person who has already been successful in a technical lead role can also be helpful in getting you to understand what they needed to get to and succeed in that position. You'll see your blind spots when you investigate your own skillset. It's funny: we accept that it takes a lot of communication and collaboration to grow and that it takes a village to raise a family, but professionally we expect that that everyone is just supposed to be able to do it all on their own. The best way to grow is to find people who have your back and to be willing to hear their feedback.

16.5.5 What's your final piece of advice to aspiring and junior data scientist?

My first piece of advice is to be *humble*: it's easy to believe that we are the kings and queens of the hill since data science is the "sexiest" profession and that any employer should be sprinkling rose petals at our feet when we walk. It's so important to remember that it takes many people to make a product successful, and that just because data science is in the limelight doesn't mean that it's better or special.

The second piece of advice is to be *kind*: it's easy to be hard on ourselves, especially because data science is so broad and can mean so many things. If you are great at analysis but not as much in machine learning engineering, you might feel that means you're not a "real" data scientist. But you are! There are so many ways to shine.

16.6 Summary

- Management is a great track for people who want to help others but are willing to give up doing data science. Following this track can eventually lead to being high up in the company.
- A principal data scientist gets to lead from a technical side and has to be responsible for others. This is a great option for staying technical while helping others more.
- Independent consulting is highly stressful and risky although potentially rewarding. You'll need a strong network of connection to consistently find work.

17

Epilogue

Well, we sure covered a lot. We started with defining data science and what skills you need to do it, walked through how to prepare for and get a data science job, and then how to grow and flourish in the field. Over sixteen chapters we went from different types of companies to making unit testing for production models to how to become a manager. Each chapter had many sub-topics each with its own lessons and themes.

Looking at the book holistically, there are a few trends that seem to flow throughout it. These are lessons that apply at all points of a data scientist's journey in different forms. For us, these three ideals have been able to keep our careers continuously moving forward.

- **A data scientist needs to be able to communicate** - Over and over, people we interviewed for the book mentioned that their success came from effectively communicating their work. Whether that's making a report for an executive to collaborating with an engineering team on a model, being able to speak in a way that non-data scientists understand will help you through the process of finding a job and with working with other people in that job.
- **A data scientist needs to be proactive** - It's exceedingly rare that a data scientist gets handed a perfectly well-formed problem and the tools to solve the problem. Instead, data scientists need to proactively try and find data, create new ideas for models, and try experiments. Being proactive and doing things like making a portfolio will also help you get a job. The more you can take initiative and find solutions to problems, the better.
- **A data scientist needs community** – No one makes it in any career without help from others, but as members of a new and fast-growing field, data scientists especially benefit from developing strong professional relationships. These relationships can take many forms. A sponsor can recommend you as a speaker for a meetup that leads two years later to keynoting an international conference. A mentor can give you feedback

on your resume and refer you to a position at their company. A manager can help bridge the gap with stakeholders and suggest areas for personal growth. Or a peer can simply commiserate with you and lift your spirits after a tough day at work. It's worth spending time building these relationships to tackle the many challenges you'll face in your career.

We hope you've enjoyed reading the book—we've certainly enjoyed writing it. In creating the work we've found so many of our own personal experiences have poured into the work. At multiple times in writing it we even reread it ourselves to better think through our own career decisions! We wish you the best of luck in your data science career journey!

Appendix A: Interview questions

Often the most helpful thing to preparing for an interview is to get into the mindset of what it will be like. Being comfortable answering questions and thinking in a way that is well suited for the fast-paced thinking of an interview can mean the different between getting the job or not. Thus, we've provided a number of interview questions for you to think about and understand. These should be viewed in conjunction with Chapter 7 on how to view the interview process as a whole. The questions in this appendix fall into five categories:

- Coding and software development
- SQL and databases
- Statistics and machine learning
- Behavioral
- Brain Teasers

This is a wide range of topics, and it's impossible to study for the thousands of questions that could be asked. One company may ask you to "invert a binary tree" while the other is only Python and behavioral questions. That's why we recommend asking before your onsite what types of questions to expect in each interview. While of course you won't get the exact questions, the hiring manager or recruiter should give you a general idea so you can focus your preparation. For example, they might say, "In your first interview you'll answer some SQL questions on the whiteboard. Then you'll have two behavioral interviews in a row, one with an engineer and one with a data scientist. Finally, one of our machine learning engineers will ask you about your previous data science projects."

It's extremely unlikely you'll only see questions from this interview during your job search process. That's why we've not only provided answers for each question (with the text what we'd say out loud, and any code what we would write on the whiteboard), but also notes on what we think makes an effective answer. For many of these questions there isn't just a single

correct answer to give; the right answer varies depending on who is asking it, how it's being asked, and the background of the interviewee. Any of our answers should be made into your own by adding your own context and opinions. The goal of this appendix isn't to provide you with all the answers in advance—it's to help you get into the mindset of an interview and think in a way that is well suited for the process.

Some of the questions come from our shared experiences going through many interviews. Others have helpfully been provided to us by people on Twitter. Thanks so much to everyone who helped making this much more useful!

A.1 Coding and software development

A.1.1 FizzBuzz

> "Write a program that prints the numbers from 1 to 100. But for multiples of three print 'Fizz' instead of the number and for the multiples of five print 'Buzz'. For numbers which are multiples of both three and five print 'FizzBuzz'."

EXAMPLE ANSWER

Here is pseudo-code for one solution to the problem:

```
for (i in 1 to 100) {
   if (i mod 15) {
      print("FizzBuzz")
   } else if (i mod 5) {
      print("Buzz")
   } else if (i mod 3) {
      print("Fizz")
   } else {
      print(i)
   }
}
```

The program iterates through the numbers 1 to 100. For each iteration it first checks if the number is divisible by 15, if so it prints "FizzBuzz". If not, it checks if the number is divisible by 5 and if so prints "Buzz". If not, it checks if the number is divisible by 3 and prints "Fizz", and if none of those are true it prints the number of the iteration.

NOTES

This problem is an extremely famous interview question in software development, having been created by Imran Ghory and popularized by Jeff Atwood, so it's common to have the exact question as part of a data science interview. The two main tasks within it are figuring out how to iterate over the set of all numbers (in the example we used a for-loop), and how to check what should be printed at each number. A common mistake is for people to first if the number

is divisible by 3 or 5 before checking if it's divisible by 15, but any number that is divisible by 15 is also divisible by 3 or 5. Thus, if 3 or 5 is checked first "Fizz" or "Buzz" might be printed in cases where "FizzBuzz" should.

While our solution proposed is straightforward, there are ways to improve on this solution. In some languages including R or Python you can take a more clean functional programming approach by using either lapply in R or list comprehensions in Python. You could also make a generalized function that as input takes the list of multiples to check and words to print at those multiples and outputs any list. Depending on how the interview is going you may want to talk about ways you would want to improve the answer.

For fun, check out FizzBuzz Enterprise Edition or a FizzBuzz TensorFlow machine learning model.

A.1.2 Tell if a number is prime

"Write a function that, given a number, returns true if it's a prime number and false otherwise."

EXAMPLE ANSWER

Here is the pseudo-code for one solution to the problem:

```
is_prime = function(n){
for (i in 2 to n / 2) {
        if (n mod i) == 0 {
      return FALSE
    }
  }
  return TRUE
}
```

A prime number is one that is not divisible by anything except 1 and itself. The program iterates through all the numbers from 2 to the half the number given and checks if the given number is divisible by them. If it is, the function returns false and stops. If it goes through the whole for loop without stopping, the function returns true.

NOTES

Similar to fizzbuzz, this problem tests if you can write a for loop and also tests if you know how to right a function. You also need to know how to stop iterating when a condition is reached, so you can safely return true at the end if the for loop completes. There are small tricks you can add like realizing you don't have to check if the number is divisible by all numbers lower than it, just those lower than its square root, but the main point of the problem is simply to write a function that works.

A.1.3 Working with git

> "Can you talk about a time when you used git to collaborate on a project?" – Alex Hayes

EXAMPLE ANSWER

At my last job, I created an R package, funneljoin, with two coworkers in an afternoon hackathon, using git from the very beginning. We spent the first hour pair programming on one computer and then made a list of tasks to split up amongst ourselves. Each of us created a different branch to work on our tasks, which enabled us to easily merge them back together in the end. Using git made sure we never accidentally overwrote someone's work. By committing early and often as we progressed, we knew we could always go back if we decided a previous way of implementing a feature was better. Finally, using GitHub meant that afterwards anyone in the company could download the package and start using it right away.

Since that initial afternoon, I've remained the maintainer of the package. I continue to use git features like branches so that I can prototype features without merging them in until they've been thoroughly tested.

NOTES

If you are asked this and haven't collaborated using git on a project before, you can talk instead about how you've used git for a personal project. One of things this question is testing is if you've used git, and if you can explain how you've used different features (like branches or forking), even if it was just by yourself. You might add about how you would adapt your practices if you were collaborating with other people; for example, by using branches more or sticking to a consistent commit message structure.

If you haven't used git before, be honest about that in the interview. But we do recommend trying to go learn it before having too many more interviews.

A.1.4 Technology decisions

> "Given a totally blank slate, how do you pick your tech stack?" – Heather Nolis

EXAMPLE ANSWER

This is an interesting question to answer because it really depends on the project at hand. My decision for a tech stack primarily relies on balancing what would be the most straightforward for me to implement with what would be the easiest for everyone else to work with. Let me give two examples of how I picked technical stacks and what I learned from them.

On one project earlier in my career, I had to develop a new product entirely from scratch and I was the only data scientist on my team. I chose to use the .NET stack and F# because I

was very familiar with them, and because I was so familiar we were able to quickly get a working product out the door. The downside was that since the language F# is so uncommon when it came time to hire a data scientist to take it over we couldn't find someone who already had the required knowledge. In retrospect .NET and F# were not the right decision.

On a more recent project I was tasked with creating a machine learning API. I was in the middle of an engineering team that worked with microservices, so I decided to create an R rest API as a docker container. While I hadn't ever used docker containers before, I chose it because I knew that would be the easiest for the team to maintain. From that project I learned a lot about docker and containers and the work I created was able to integrate well.

NOTES

When interviewing candidates at T-Mobile, Heather usually has a person pick a project and describe the decisions they made, the decisions other people made, and how they would do it differently knowing what they know now. Your interviewer may not ask you those things directly, but they are all worth including anyway.

Whatever answer you give you'll want to include plenty of references to decisions you have had to make in work you've done before (which can include side projects or course work). The point of this question is to see how much thought you have put into choosing the right technology for the right project. Having chosen technological stacks that ended up being problematic is fine as long as you've learned from it. In fact, having learned from things is even better than getting everything right the first time because it shows you can change.

A.1.5 Frequently used package/library

"What's an R package or Python library you use frequently and why?"

EXAMPLE ANSWER

It's not one package, but I really love the suite of packages that make up the Tidyverse in R. The packages get you all the way from reading in the data, to cleaning it, to transforming and visualizing and modeling it.

I especially enjoy working in dplyr because thanks to the connected package dbplyr, I can write the same code whether I'm working with a local or remote table as dbplyr translates dplyr code to SQL. Using this at my last job meant I could stay in RStudio for my entire workflow even though all of our data was stored in redshift and required SQL queries. I would use dbplyr to do summaries and filtering and then pull the data down locally if I needed to do more complicated operations or visualize it.

Overall, I really like the philosophy of Hadley Wickham, a core tidyverse developer: that the bottleneck is often thinking time, not computational time, and that you should build tools that work seamlessly together and let you translate your thoughts to code quickly.

NOTES

The interviewer shouldn't be looking for a specific answer here. Rather, they're looking to see if you a) program enough in either language to have a frequently used package and b) can explain how and why you use that package. This answer also gives them a sense of what kind of work you do day-to-day. Don't forget to explain what the package does, especially if it's a niche package. If there's another package that's more widely used for the task, you may want to explain your reasoning of why you choose this package, as that shows your broader awareness of what alternatives are out there. Finally, don't worry about picking the most "advanced" library (like a deep learning library) to impress the interviewer. This should hopefully be one of the easiest and potentially fun questions to answer, so don't overthink it.

A.1.6 RMarkdown or Jupyter Notebooks

> "What is an RMarkdown file or Jupyter Notebook? Why would you use an RMarkdown file or Jupyter Notebook over an R or Python script? When is a script better?"

EXAMPLE ANSWER

I'm going to answer this in the case of R and RMarkdown, but the basic idea is the same for Python and Jupyter Notebooks. RMarkdown files are ways of writing R code that allow you to put text and formatting around the code. In a sense they merge the code and results of an analysis with the narration and ideas of the analysis. By using an RMarkdown file you can have an analysis that is easier to reproduce than raw R code with separate documentation on what the analysis was. Ideally, your RMarkdown would be formatted so clearly that when you render the output file you could hand the resulting HTML output, word document, or PDF to a stakeholder.

RMarkdown files are great for reproducible analysis, but they're less useful when you're writing code you're going to deploy or use in other places. Say you have a list of functions that you want to use in multiple other places (such as one to load the data from a file). It might make sense to write an R script that creates all the functions and keep that separate from an individual analysis. Or if you wanted to use R with the Plumber package to create a web API you wouldn't want an RMarkdown file for that.

NOTES

This question is a check from the interviewer to see if you have experience in doing reproducible analysis. Many people who use R or Python write scripts in ad hoc ways and don't think about how they will share the results with others. By showing you understand the point of RMarkdown and Jupyter Notebooks you display that your thinking about how to make your code more usable. If you're reading this and you haven't used either RMarkdown files or Jupyter Notebooks then definitely go try either of them out.

Don't worry about understanding both R and Python versions of this, one should be fine.

A.1.7 When should you reuse code?

> "At what point should you reuse code as a function? When should you turn it into a package or library?"

EXAMPLE ANSWER

In general, if I notice that I am ever copying and pasting code it's probably a sign I should make a function out of it. For instance, if I need to run code on three different data sets, I should make a function and apply it to each one rather than copying the code three times. The library purrr in R or list comprehensions in Python make it easy to apply a function many times.

I have found that packages and libraries are best when you have code that spans multiple distinct projects on the team. For instance at my current job we have a lot of data that we store in S3 but want to analyze locally. Rather than copying and pasting functions to access the code in each project, I created a library that could be called from all of them. The downsides of libraries is that if you change them then you have to change all of the projects that use the library, but for core functions this is often worth it.

NOTES

This is somewhat of a softball question because it has a right answer: "as much as possible." Generally copying and pasting code over and over is a bad practice and a data scientist should make functions so that the code is easier to read and understand. Because of that, the more you can add examples that show you understand this the better. Was there a time you made a function and reused it a lot? A library? Talk about those situations as much as possible.

A.1.8 Example manipulating data in R/Python

"Here's a table, called tweets. The data has the account that sent the tweet, the text, the number of likes, and the date sent. Write a script to get a table with one row per person with a column that's the minimum number of likes they got, called `min_likes`, and a column of the total number of tweets, called `nb_tweets`. This should only be for tweets sent after September 1st, 2019. You also need to eliminate any duplicates in the table first."

account_name	text	nb_likes	date
@vboykis	Data science is...	50	2019-10-01
@Randy_Au	It's hard when...	23	2019-05-01
@rchang	Some news...	35	2019-01-01
@vboykis	My newsletter...	42	2019-11-23

@drob	My best advice...	62	2019-11-01
...	...	...	...

EXAMPLE ANSWER IN R

```
tweets %>%
filter(date > "2019-09-01") %>%
distinct() %>%
group_by(account_name) %>%
summarize(nb_tweets = n(), min_likes = min(nb_likes))
```

EXAMPLE ANSWER IN PYTHON

```
(tweets[(tweets['date'] > '2019-09-01')].
drop_duplicates().
group_by('account_name').
agg({'nb_tweets': 'count', 'min_likes': 'min'})
)
```

NOTES

This type of question is a mix between the fizzbuzz and prime questions (knowing how to do something in R/Python) and SQL queries (analyzing data). This specific question should be relatively easy for someone who has done data analysis before, but you may face one that has tricks (like needing to convert a character column to a date column or change it from long to wide format) that you can't remember in the moment. If you don't remember how to do something, just say "I don't remember the exact syntax for X, so I'm going to put some pseudocode in there now as a placeholder" and move on. You don't want to spend too much time getting stuck on one part. If it does include something that's more unusual, it's likely the interviewer considers that part a "bonus" rather than a requirement for going on to the next stage.

A.2 SQL and databases

A.2.1 Types of joins

> "Explain the difference between a left join and an inner join" – Ludamila Janda and Ayanthi G.

EXAMPLE ANSWER

Joins are ways of combining data from two different tables, a left table and a right table, into a new one. Joins work by connecting rows between the two tables—a set of key columns are used to find data in the two tables that are the same and should be connected. In the case of

a left join, every row from the left table will appear in the resulting table, but rows from the right table will only appear if the values in their key columns showed up in the left table. In an inner join however, both rows from the left table and the right table will only appear if there is a matching row in the other table.

In practice, you can think of a left join ask attaching data from the right table to the left if it exists (like using the right table as a lookup). An inner join is more like finding all the shared data and making a new table from only the pairs.

NOTES

Janda likes this question as an early screener for more junior roles since it isn't a trick and is an important knowledge for a candidate to have. She finds you can learn a lot from how the candidate chooses to answer. There are plenty of valid answers from ones that are textbook correct but not easy to understand to very simple to understand ones that miss edge cases.

Notice for instance that in our answer we didn't talk about any complexities from duplicate rows appearing in the data. It might be work mentioning it since that can affect results, but more likely it's a distract from the point you're trying to get across.

A.2.2 Loading data into SQL

> "What are some different ways you can load data into a database in the first place, and what are the advantages and disadvantages of each" - Ayanthi G.

EXAMPLE ANSWER

There are many ways to load data into a database, primarily depending where the data exists in the first place. If the data is in a flat file such as a CSV file, many SQL versions have programs to import the data. For example, SQL Server 2017 has an Import and Export wizard. These tools are easy to use but do not allow for much customization and are not easily reproducible. If the data is coming from a different environment like R or Python, there are drivers that allow for passing the data into SQL. For instance, an ODBC driver can be used along with the DBI package in R to move data from R into SQL. These methods are more reproducible and programmatic to implement, but require you get the data into R or Python.

NOTES

This question is really a test of whether you've had to load data into a database before or not. If you've done it before then it shouldn't be difficult to describe how you did it. If you haven't loaded data into a database before that might be a signal to an interviewer that you don't have enough experience.

The part of the question about advantages and disadvantages is to check that you understand that different tools are better in different situations. Sometimes using a GUI to upload data is a nice easy solution when you have a single file. Other times you'll want to set

up a whole automated script to load the data continuously. The more you can show you understand the nuance of what to use when, the better.

A.2.3 Example SQL query

> "Here is TABLE_A from a school containing the grades, from 0 to 100, earned by students across multiple classes. How would you calculate the highest grade in each class?"

TABLE_A – a table containing the grades from each student in each class at a school

CLASS	STUDENT	GRADE
Math	Nolis, Amber	100
Math	Berkowitz, Mike	90
Literature	Liston, Amanda	97
Spanish	Betancourt, Laura	93
Literature	Robinson, Abby	93
…	…	…

EXAMPLE ANSWER

Here is a query to find the highest grade in each class:

```
SELECT CLASS, MAX(GRADE)
INTO TABLE_B
FROM TABLE_A
GROUP BY CLASS
```

This query groups the data into each class, then finds the max from it. It additionally saves the result into a new table (TABLE_B), so that the results can be queried later.

NOTES

This question is almost as simple as you can get for a SQL question–it's testing that you have a basic understanding of grouping in SQL. The reasons people typically mess this question up is they either don't correctly see what to group (in this case the class variable), or they find the question so easy that they overcomplicate it in their head and miss the simple solution. If you are in an interview and a question seems too easy it very well may be as easy as it seems.

If you're reading this and that solution didn't seem obvious, it would be a good idea to review how grouping variables work in SQL. Lastly, the INTO TABLE_B line was totally optional, but it sets us up well for the next question.

A.2.4 Example SQL query continued

> "Consider the table from the previous question. What if we wanted to not only find the highest grade in each class, but also the student who earned that grade?"

EXAMPLE ANSWER

Assuming we have the result from the previous question stored in TABLE_B, we can use it in the solution here:

```
SELECT a.CLASS, a.GRADE, a.STUDENT
FROM TABLE_A a
INNER JOIN TABLE_B b ON a.CLASS = b.CLASS AND a.GRADE = b.GRADE
```

This query selects all of the students and their grades from the original TABLE_A that have classes with grades that show up in the table of maxes, TABLE_B. The inner join acts as a filter to keep only the class/grade combinations that are the maxes, since only in that case does the grade appear in TABLE_B. Alternatively, we could use a subquery to do the same thing without calling TABLE_B:

```
SELECT a.CLASS, a.GRADE, a.STUDENT
FROM TABLE_A a
INNER JOIN (
        SELECT CLASS, MAX(GRADE)
        FROM TABLE_A GROUP BY CLASS) b
ON a.CLASS = b.CLASS AND a.GRADE = b.GRADE
```

NOTES

While there are multiple solutions, any solution almost certainly requires more than just a single query off of TABLE_A, and because of that this question can easily trip people up. While the solution may appear straightforward on paper, being able to think of it during an interview might be hard—this is the kind of question that if you get wrong it'll still be possible to pass the interview.

The solution doesn't make any special case for if there is a tie for the max—in the case of the example solution multiple students would be returned. It might be worth pointing this out to the interviewer since it shows you're paying attention to edge cases.

A.2.5 Data types

> "What disadvantages are there to storing a column of dates as strings in a database. For example, in SQL what if we stored a column of dates as a VARCHAR(MAX) instead of DATE?"

EXAMPLE ANSWER

Having dates stores as strings instead of dates (such as storing March 20th, 2019 as the string "03/20/2019") is a common situation in databases. While depending on how you do it you may not lose any information, there are performance hits from this. First, if the data isn't stored as a DATE type, then we couldn't use the MONTH() function on it. We also couldn't do things like find the differences between two dates or find the minimum date in the column.

This problem tends to happen a lot when loading data into a database or cleaning it. The earlier you can correctly format the data, the easier the analysis will be. You can fix these sorts of situations using functions like CAST. That being said, if you're loading in data with hundreds of columns and there are plenty you'll never use, it may not be worth the time to fix all these issues.

NOTES

This sort of problem of having data stored in an incorrect type is an extremely frequent occurrence. It doesn't just happen in databases; it can also happen in flat files or in tables within environments like R or Python. This question is checking you understand that when this happens it's generally bad, and when you see these situations you generally should fix them. Being able to answer questions like these should naturally come from doing data cleaning as part of data science projects that start with messy data.

A.3 Statistics and machine learning

A.3.1 Statistics terms

> "Explain the terms 'mean', 'median', and 'mode' to an 8-year old." - Allan Butler

EXAMPLE ANSWER

Mean, median, and mode are three different types of averages. Averages let us understand something about a whole set of numbers with just one number that summarizes something about the whole set.

Let's say we did a poll of your class of how many siblings each person has. You have five people in your class, and let's say you find one person has no siblings, one has one, one has two, and two have five!

The mode is the most common number of siblings. In this case that's five, as two people have five siblings compared to only one person having every other number.

To get the mean, you get the total number of siblings and divide that by the number of people. In this case, we add 0 + 1*1 + 1*2 + 5*2 = 13. You have five people in the class, so the mean is 13/5 = 2.6.

The median is the number in the middle if you line them up from smallest to largest. We'd make the line of 0, 1, 2, 5, 5. The third number is in the middle, and in our case that means the median is two.

We see the three types of averages all come up with different numbers. When do you want to use one instead of the other? The mean is the most common, but the median is helpful if you have outliers. For example, imagine one person had 1000 siblings! Suddenly your mean gets much bigger, but that doesn't really represent the number of siblings most people have. On the other hand, the median stays the same.

NOTES

It's unlikely that someone interviewing doesn't know about the different types of averages, so this question is really looking to test your communication skills rather than if you get the definitions right (although if you get them wrong that's a red flag). In our example, we used a simple example an eight-year old might encounter in real life. We recommend keeping the number of subjects simple; you don't want to get tripped up doing the math for the mean or median because you're trying to calculate them for fifty data points. As a bonus, you can add as we did when you might want to use one instead of another.

A.3.2 Explain p-value

"Can you explain what a *p*-value is to me and how it's used?"

EXAMPLE ANSWER

Let's imagine you were flipping a coin and got 26 heads out of 50. Would you conclude it wasn't a fair coin because you didn't get exactly 25 heads? No! You understand there's randomness at play. But what if it came up heads 33 times? How do we decide what the threshold is for concluding it's not a fair coin?

This is where the p-value comes in. A p-value is the probability that, if the null hypothesis is true, we'd see a result as or more extreme than the one we got. A null hypothesis is our default assumption coming in, such as there's no differences between two groups, that we're trying to disprove. In our case, it's that the coin is fair.

Since a *p*-value is a probability, it's always between 0 and 1. The *p*-value is essentially a representation of how surprised we would be by a result if our null hypothesis is true. We can use a statistical test to calculate the probability that, if we were flipping a fair coin, we would get 33 or more heads or tails (both being results that are as extreme as the one we got). It turns out that probability, the *p*-value, is .034. By convention, people use .05 as the threshold for rejecting the null hypothesis. In this case, we would reject that the hypothesis that the coin is fair.

With a p-value threshold of .05, we're accepting that 5% of the time, when the null hypothesis is true, we're still going to reject it. This is our false positive rate: the rate of rejecting the null hypothesis when it's actually true.

NOTES

This question is testing two things: whether you understand what a *p*-value is and if you can communicate that and how it's used. There are common misconceptions about the *p*-value, like that it's the probability a result is a false positive. Unlike the averages question above, it is possible someone gets this wrong. On the communication side, we recommend using an example to guide the explanation. Data scientists need to be able to communicate with a wide variety of stakeholders, some of whom have never heard of *p*-values and some who think they understand what they mean but don't. You want to show you both understand *p*-values and can share that understanding with others.

A.3.3 Explain a confusion matrix

"What's a confusion matrix? What might you use it for?"

EXAMPLE ANSWER

A confusion matrix lets you see for a given model how your predictions compare to the actual results. It is a 2x2 grid has four parts, the number of true positives, false positives, true negatives, and false negatives. From a confusion matrix, you can calculate different metrics like accuracy (the percent classified correctly as either a true positive or true negative) and sensitivity, otherwise known as the true positive rate, the percent of positives correctly classified as such. Confusion matrixes are used in supervised learning problems, where you're classifying or predicting an outcome, like whether a flight will be late or if a picture is of cat or a dog. Let me draw an example one for the flight outcomes:

	Actual late	Actual on-time
Predicted Late	60	15
Predicted on-time	30	120

In this case, 60 flights that were predicted to be late actually were, but 30 predicted to be on-time were actually late. That means our true positive rate is 60 / (60 + 30) = 2/3.

Seeing the confusion matrix instead of just a single metric can help you understand your model performance better. For example, if you just calculated the accuracy, you might find that you have 97% accuracy. That sounds great, but it turns out that 97% of flights are on time. If the model simply predicted that every flight is on-time, it would have 97% accuracy, as all the on-time ones are classified correctly but would be totally useless!

NOTES

This question tests if you're familiar with supervised learning models. It's also testing if you know different ways of evaluating the performance of models. In our answer, we shared

two metrics you could calculate from a confusion matrix, showing we understand how it could be used, and also a case where seeing the whole matrix instead of just one metric is useful.

A.3.4 Interpreting regression models

> "How would you interpret these two regression model outputs given the input data and model?" This model is on a dataset of 150 observations of 3 species of flowers: setosa, versicolor, and virginica. For each flower, the sepal length, sepal width, petal length, and petal width is recorded. The model is a linear regression predicting the sepal length from the other four variables."

```
 Sepal.Length Sepal.Width Petal.Length Petal.Width Species
         <dbl>       <dbl>        <dbl>       <dbl> <fct>
1          5.1         3.5          1.4         0.2 setosa
2          4.9         3            1.4         0.2 setosa
3          4.7         3.2          1.3         0.2 setosa
4          4.6         3.1          1.5         0.2 setosa
5          5           3.6          1.4         0.2 setosa
```

Input data to the model

```
model <- lm(Sepal.Length ~ ., iris)
```

Model call

```
term              estimate std.error statistic  p.value
<chr>                <dbl>     <dbl>     <dbl>    <dbl>
(Intercept)          2.17     0.280       7.76 1.43e-12
Sepal.Width          0.496    0.0861      5.76 4.87e- 8
Petal.Length         0.829    0.0685     12.1  1.07e-23
Petal.Width         -0.315    0.151      -2.08 3.89e- 2
Speciesversicolor   -0.724    0.240      -3.01 3.06e- 3
Speciesvirginica    -1.02     0.334      -3.07 2.58e- 3
```

Output 1

```
variable       value
<chr>          <dbl>
r.squared      0.867
adj.r.squared  0.863
sigma          0.307
statistic      188
p.value        2.67e-61
df             6
logLik         -32.6
AIC            79.1
BIC            100
```

```
deviance        13.6
df.residual      144
```

Output 2

EXAMPLE ANSWER

Looking at the model summary results, it looks like it's a very good model; the r-squared is 0.867, meaning that the predictors explain 86.7% of the variance in sepal length. The predictors are all significant at the p less than .05 level. I see that the wider the sepal and the longer the petal, the longer the sepal, while wider petals actually are associated with shorter sepals. Both the versicolor and virginica species have negative coefficients, which means that we'd predict those species to have a smaller sepal length than the setosa species.

Let's say we found a new flower, with a sepal width of 1, petal length of 2, petal width of 1, and it was the virginica species. Our model would predict the sepal length to be the following: 2.17 + .496 * 1 + .829 * 2 - .315 * 1 – 1.02, which is about 3. Before using this model though, I'd want to look at a few more diagnostics, like whether the residuals are normally distributed, and I'd want to find a test set to see how it performs out-of-sample to make sure it's not overfitting.

NOTES

This question is one where the interviewer is looking for multiple things, and you can get "points" depending on how many you get, rather than being a question where you either get it right or wrong. In this case, the interviewer is looking that you understand the model statistics (like *r*-squared) and the estimates and their associated p-values. While not explicitly asked for, in our answer we also added how we would use this model to predict the sepal length given a new flower. Finally, we added some more information about the model we'd want to know before we started using it. This type of open-ended question is a good opportunity to hit what the interviewer is probably looking for and to add on "bonus" information. Avoid trying too hard and ending up spending 20 minutes on a single question—show that you understand as many concepts as you can then move on.

A.3.5 What is boosting?

> "What does the term 'boosting' mean when referring to machine learning algorithms?"

EXAMPLE ANSWER

"Boosting" refers to a whole class of machine learning algorithms that are built on taking a weak model and reusing it enough times that it becomes a strong one. The idea is you train a weak model on data, then you look for areas where the model had errors and train a second model that weights the datapoints where there were errors more heavily hoping the second

model will fix some of the mistakes of the first. You repeat this again and again until you hit some limit to the number of models. Then you use all of these models together to make the prediction. By having a large set of models you'll get a much more accurate result than just having used a single model. One very popular implementation of a boosting method is XGBoost, which is used a lot in both R and Python.

NOTES

Boosting is an uncommon enough term that is entirely possible someone with a basic data science background might not know exactly what it means. Thus, this question is more of a test of something being senior more than someone having any data science expertise. It's also a little bit academic—you can imagine someone having successfully used XGBoost for years in their code without thinking too deeply of exactly how it works. This question should be thought more of a "it's nice to get it right but not the end of the world" than a "if you don't get this question right you're unlikely to get the job."

A.3.6 Favorite algorithm

> "What's your favorite machine learning algorithm? Alright, can you explain it to me?" - Jeroen Janssens

EXAMPLE ANSWER

My favorite machine learning algorithm is a recurrent neural network. I have been doing a lot of work lately with natural language processing, and recurrent neural networks are great models for quickly classifying text.

Do you know a linear regression? A neural network is like a linear regression except you have a whole bunch of linear regressions and the output of one group of linear regressions is the input to the next group of linear regressions. By tying all these linear regressions together into layers of models, you can much more accurately make predictions. A *recurrent* neural network is a special case of a neural network that is tuned for data that falls in sequences. In the case of natural language processing a block of text, the output part of the way through a sequence of words is the input for the model of the next words.

NOTES

This question is one of the many questions you might see during an interview that are designed to see if you can explain a complex idea in a simple way. What algorithm you choose for your answer isn't nearly as important as being able to express how it works clearly. That said this is a great opportunity for you to talk about interesting past work you've done by expressing an algorithm that relates to the work and talking about it.

A.3.7 Training vs test data

> "What is training data and what is test data? What is your general strategy for creating these datasets?"

EXAMPLE ANSWER

Training data is data that is used to train a machine learning model. Test data is data that is not used in training a machine learning model, and instead is used to validate how well the model works. These datasets need to be separate because if data is used to train a model, then the model can learn the correct result for the data and will be artificially good at fitting to it.

There are many ways to split training data and test data. My general approach is to take a small random sample, such as 10%, at the beginning of an analysis and using that as my test data for all my models while the other 90% is training data. Once I've found a model I like and performs well enough, I retrain the model on all the data (both training and test) to get the most accurate model to deploy to production.

NOTES

It's really important that you have a good explanation ready for the difference between training and test data, because understanding the distinction and how to think about them is a fundamental part of creating a machine learning model. That being said, there are many valid strategies for splitting your data. For instance besides just randomly sampling you could use cross validation to avoid biasing your model while training it on more data. So long as you have a logical explanation for why you choose a method you should be good.

A.3.8 Feature selection

> "How would you do feature selection if you had 1,000 covariates and had to reduce it to twenty?" – Alex Hayes

EXAMPLE ANSWER

There are a number of different ways to do this. One possible solution in the case that it's a prediction problem would be to use a lasso regression. A lasso regression is a special type of linear regression that applies a penalty to increasing the value of the coefficients. By increasing the penalty term in the regression you can make the model have fewer and fewer coefficients, until it uses only the twenty most important covariates. In this way the model selects what the coefficients should be in the model. While a lasso regression will have a lower accuracy score than a linear regression with all the covariates on the training data, it has the benefit of only using a small number of them.

You could also use dimensional reduction techniques like Principal Components Analysis (PCA) to reduce the dimensionality of the problem from 1,000 to 20. The lasso approach will choose 20 features out of the existing 1,000. Methods like principal components analysis will create 20 new features that try and capture as much of the data from the 1,000 as they can.

NOTES

There are many possible solutions to this. One more solution could be to try and use a stepping function to remove covariates over and over until you're down to twenty. You could even take many samples of sets of 20 features and choose the set that works the best. Think of this question as less of a test of knowing the exact right approach to a problem, and more you being able to show that you could find a solution if you were faced with this problem. This question is a test to make sure you won't get stuck on the job—can you think of anything you would want to try? If yes, great you can go try it, if not then you may struggle when working on your own.

A.3.9 Deploying a new model

> "You developed a new model that performs better than your old model currently in production. How do you determine whether you should switch the model in production? How do you go about it?" – Emily Spahn

EXAMPLE ANSWER

For me, this depends on a couple of factors of the environment. First, by what metric does the new model do better? Assuming it's overall accuracy, I would check that the model is sufficiently better that it's worth swapping out the old model. If it's only a percentage point better in accuracy, it might not be worth the effort of changing since that might be a negligible effect. Next, is there a risk to disrupting the current model? If the model was deployed using a well maintained pipeline with clear logging and testing then I would probably make the swap, but if the model was say deployed by manually moving a model into a production system by a person who is no longer at the company, I would probably hold off.

Lastly, is there a way to A/B test the model first? Ideally, I would like to concurrently have the old and new models run in parallel so I could test that there aren't any problems with the new model or edge cases missed by it. No test system can cover everything from production, so being able to have it running for a select set of customers or inputs first would be ideal.

NOTES

Deploying a model is often a labor intensive and risky proposition for a company. This question is determining if you understand what that's like and how you would approach the situation. A more junior data scientist or machine learning engineer may feel like the right choice is to deploy the most accuracy model as quickly as possible, but there are risks that

need to be managed. If you have any past experiences you can draw on (like model deployments failing) this is a great question to mention them in. If you haven't that's totally fine, just try and describe what you think might go wrong.

A.3.10 Model behavior

> "Given a model you developed, how would you design a metric to evaluate it from the end user perspective? How would you decide what errors are acceptable or not?" - Tereza Iofciu and Bertil Hatt

EXAMPLE ANSWER

Standard model metrics like R^2 or accuracy can miss the end user or business perspective. For instance, a classification model could be right 99% of the time, but the 1% of the times it's wrong it's such a problem for the business that the model would never be used.

I find that the best way to evaluate a model is to try and do an experiment with it. For example, if I'm creating a model to cluster customers into segments then I would present the clusters to marketing and have them try and do a test run of doing custom marketing to a sample set of customers from the different segments. If having that segmentation data means that the marketing is more effective than the model is good. Or in the case of a predictive model if the business is able to use those predictions effectively and increase their metrics than the model is value. That is more meaningful than any statistical metric about the model itself.

NOTES

This is a tricky question because it's very general but to answer it you need to talk about specifics. Your answer could vary dramatically for a predictive model vs an unsupervised model, or if you're working with marketing vs the operations department. You'll want to talk a lot about the idea that the statistical measures aren't the same as the measures that the business cares about—junior data scientists can often get overly focused on just maximizing the statistical measures and ignoring the business ones. But how you end up talking about these ideas is very open. Like many questions, if you can bring examples from your past experiences it can add a lot of depth to it.

A.3.11 Training and test sets

"Why do you generally split data into training and test sets for machine learning problems?"

A.4 Behavioral

A.4.1 Project that had the most impact

"What's the project you worked on that had the biggest impact?"

EXAMPLE ANSWER

In my last job, I was brought in to build their online experimentation, or A/B testing, analytics system. The company was interested in starting to run experiments and had an engineer who could implement them, two growth marketers who could come up with the ideas and lay out the changes, and a manager, but they needed a way to understand what the results of the experiment were. When I first started, I analyzed each experiment individually in R. But I knew this wasn't the best system: it meant that the team needed me to run the scripts to see the results and that I was duplicating work across analyses.

That's what led me to build an internal dashboard to monitor experiments. This dashboard included not only the results of each experiment, such as the percentage of people who registered or subscribed in the control versus the treatment group, but also health checks to make sure the experiment was running as expected and the results could be trusted. With this dashboard, anyone at the company could check and see the most up-to-date results.

By the time I left, this dashboard was being used for all the experiments being run across five different teams. Thanks to the work I did with the rest of the experimentation team, almost every feature the company launches now is first tested as an experiment to measure if it has any positive impact.

NOTES

For this answer, if you have done any data science projects for a company, you want to use one of those instead of a non-data science project. On the other hand, if you've only done data science projects for personal use or class, you can highlight another project. The biggest thing here is to focus on the impact to the business. Saying "I built a model with 90% accuracy!" is not what they're looking for; they want to understand how someone used the model, tool, or analysis you built and why it mattered.

A.4.2 Data surprises

"Can you tell about a time you found something in the data that surprised you?"

EXAMPLE ANSWER

My previous job was at a company that made money with subscriptions. I worked on experiments there and when I first started, I would calculate the subscription rate in an

experiment as the percent of people who entered who later subscribed. While that sounds good, I found out soon that it turns out people had subscriptions starting in the future!

After talking with the data scientist who owns the subscription data, it turns out that these subscriptions with future start dates were subscriptions that someone had paused. For example, let's take a user with a monthly subscription starting in September. She could choose instead of renewing or canceling her subscription in October to pause it, not paying and losing access for two months but then having the subscription start again in December. In this case, she would have two rows in the subscription table: one for September to October subscription and then another starting in December.

For my use-case, I didn't want to count those subscriptions that would start because they would become un-paused; I only wanted subscriptions where someone was actively choosing to subscribe! I learned two lessons here: that I should never make assumptions about the data and that I may need to customize a data source for my needs. I had assumed that it would be impossible for subscriptions to start in the future and so hadn't checked for that. When I realized this issue, I didn't overwrite the original data, since other people still needed to know about subscriptions that had been set to start in the future. Instead, I made my own table that only counted new subscriptions.

NOTES

In this answer we used an example where we were surprised by what is essentially a data quality issue for our use-case. But you could answer instead about a time your intuition just didn't match with the results; for example, an exploratory data analysis you did of the reddit sub-thread on data science where you thought the word count of posts would correlate positively to the number of comments but it turned out there was a negative correlation. You also want to make sure you explain why you held your initial assumption.

This question is looking to see that you think about your data before simply diving into it. It also is testing that you don't just try to confirm your initial hypothesis but rather let yourself be surprised by results and adapt given the new information.

A.4.3 Previous job reflections

> "What is the thing that you wanted to change most in your previous job that you couldn't?" - Bertil Hatt

EXAMPLE ANSWER

I found that at my last company there were real struggles with communication. The leadership team was constantly asking people to be more open and express their concerns, but it wouldn't happen. My theory is that it was due to the leaders themselves not being open—they would constantly tell us everything was going great when we knew that there were problems. The thing I wanted to change the most is to have the leadership open up to us. If they would

express their own struggles and concerns more it would have made it easier for the more junior employees to open up and for a better working environment.

NOTES

This is a *tricky* question. You basically need to show that you understood your previous working environment well enough to have a proposed improvement for it, but you need to do it while making it sound like you had a good relationship with your previous employer.

There are lots of different types of changes you could list, things like: technical changes, team dynamic changes, and product changes. The more meaningful of a change you could list the better (so no "I wish we had free soda"). It's also great if you can reflect on why that change didn't happen ("I wish we had been using a modern language like R or Python, but we were using SAS because of all the legacy products").

Avoid insulting your previous employer (like: "can you believe they use FORTAN??"). You don't want to give the impression that someday you'll leave your next company and insult them too. Be respectful of the work your past employer has done, even if it's ultimately lacking.

A.4.4 Senior person making a mistake based on data

> "What would you do if you had calculations or results that conflicted with the previous results of a senior person in the company. Would you try and convince them you were right, and if so how?" - Hlynur Hallgrímsson and Heather Nolis

EXAMPLE ANSWER

First, I'd ask myself if this important enough to bring up. If they were off by a small percent but we'd still make the same decision with the new results, or the previous results were never actually used for anything, I might let it go.

If not, I'd start by trying to understand the motivations and goals of the other person. For example, let's say this was the VP of Sales and they had done an analysis showing that each sales person they hired brought in more than twice their salary in sales. They then used this to justify hiring five more people for the team. If I show that actually each sales person is bring in less than their salary, that could jeopardize the whole sales department. This is a very different situation than if it was a small analysis they did for a side project.

I would then set up a meeting with them. By understanding the situation, I can make an educated guess for how they will react. In the case where the results conflicted because they had an error in their analysis or it was a fundamental result to their business, I'd expect they might be defensive and try to find flaws in my analysis, so I'd prepare emotionally and by triple-checking my results. I would try to find a solution that lets them save face, pivot their strategy, and put the business in the right direction.

In the worst case where they didn't listen, didn't offer any valid reasons why the new results were wrong, and I think the new results are vital to the business, I'd work with my manager to come up with a strategy to have the new results shared and acted on. Unfortunately, sometimes people won't end up agreeing with your new analysis, and the focus needs to shift from convincing them to finding a way to accomplish your goals and limit the impact of a wrong analysis.

NOTES

This question is looking to understand how'd you handle conflict with someone more senior. While many different answers can work, "I would email everyone in the company to talk about how wrong and stupid they were" or "I'd always handle it this way no matter the situation because the only thing that matters is the data" would definitely be an issue. Academics can especially struggle with handling conflict in businesses; in academia, talks can be a contest of who in the audience can find the most flaws in the research and tear down the arguments. In industry on the other hand, you need to be able share your viewpoint and help business make right decisions while balancing other factors and understanding the nuances of different situations. The interviewer is looking for signs that you've successfully resolved disagreements before so exhibit that as much as you can.

A.4.5 Disagreements with teammates

> "Tell me about a time you disagreed with a teammate. What was it about and what did you do?"

EXAMPLE ANSWER

One time I was working with a product manager on an experiment where we'd set a runtime of two weeks based on a power calculation. Four days in, they wanted to stop the experiment early and fully launch it because the p-value was .04 on the main success metric. But I knew this could be an artifact of peeking: by checking your results every day to see if the p-value dips below .05 and stopping if it does, you greatly increase your false positive rate. I also knew the product manager was highly incentivized to have a successful experiment: one of the main metrics they were evaluated on was the incremental revenue gained through successful experiments.

In this case, I focused on making sure I knew where they were coming from and asking questions. I brought us back to our shared goals: making the company as successful as possible. I walked them through a simple example from XKCD to help them develop the intuition of why stopping early could be a problem: that if you checked if 20 different colors of jelly beans were associated with acne, even if none were, by chance a statistical test would likely "find" that one of them was. In the same way, we were "chasing statistical ghosts" and were liable to trick ourselves into thinking we had a positive impact when we didn't. In the end, they agreed to keep the experiment running for the planned two weeks.

This situation also led to me thinking more about how I could improve the experimentation tool to make it easier for people to do the right thing. For example, one experimentation platform I know has a little circle that fills up more every day and at the end of 7 days turns in to a checkmark. This helped people run their experiments for at least a week, which is best practice.

NOTES

This answer uses the STAR approach: situation, task, approach, result. This is a classic framework for answering behavioral interview questions, as it provides a structure for the answer that is easy to follow. When thinking of a good example for this question, you want to find a situation that had a positive result, not "and then we never spoke again" or "I got him fired." You also want the disagreement to be business-related not "we disagreed how to load the office dishwasher." Interviewers here are looking to see if you can empathize with someone you disagree with and avoid bad-mouthing them or blaming others for your problem.

A.4.6 Difficult problems

> "What do you do when you don't know how to solve a data science related problem?"

EXAMPLE ANSWER

For coding questions, Google is my friend - often I'll find the answer in the Stack Overflow question that is the first result if I google an error message or something like "how do I do LDA in R?" If I know what the function or package I want to use is but I'm not sure exactly how it works, I'll check if there's any documentation.

If Google and documentation fail me, I've found Twitter very helpful. For example, I wanted to know how to set a global color theme for my plots but google wasn't returning any results. I asked on twitter with the #rstats hashtag and someone responded by pointing me to the ggthemr package and the blog post they'd written about how to use it.

I also like the rule of spending 15 or 30 minutes on the problem myself (depending on if I feel I'm making any headway) and then asking another data scientist at my company for help. It's my responsibility to first try to figure it out on my own, but it's also on me not to be silently stuck on something for a whole day when a colleague could have helped me in a few minutes. When I reach out, I'll both share what I've tried and also a small, reproducible example to make it easier for the other person to see the issue (rather than sending them hundreds of lines of code to parse through).

NOTES

Data science is a field where you'll be constantly learning and challenged by problems you've never seen before, so it's important that you develop a few strategies for getting unstuck. One

thing this question is looking for is that you've developed strategies for outside of the classroom setting where you have an answer sheet, classmates, and a professor all there to help you. You'll potentially need to tailor this answer to the company you're talking to: if you say your main strategy is asking your data scientists colleagues, and you're interviewing to be the first data scientist, that will be a red flag.

A.5 Brain teasers

A.5.1 Estimation

> "What's an estimate for how many mini shampoo bottles are used by all the hotels in the United States in a year?"

EXAMPLE ANSWER

Let's estimate the number of bottles using the following formula:

*number of hotels in the US * average number of rooms per hotel * 1 shampoo bottle per occupied room per night * average occupancy per room * 365 days per year = number of shampoo bottles per year*

Then let's estimate the numbers in the formula

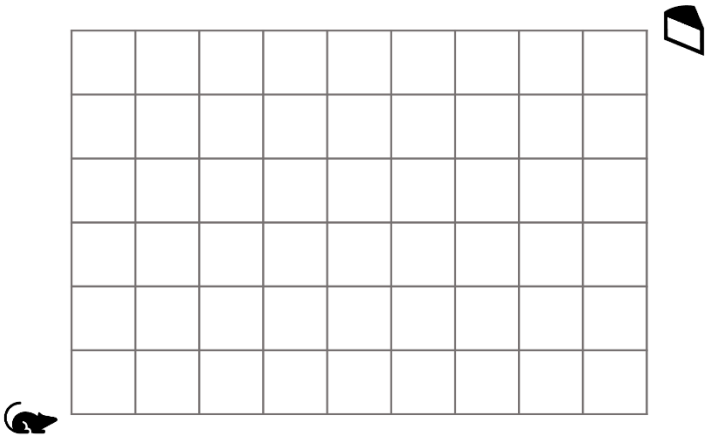
- Number of hotels in the US – If we assume that there is a hotel for every 5000 people in the country, and there are around 300 million people in the country, that's 60,000 hotels
- Number of rooms per hotel – 50 seems like a decent guess for the average number of rooms in a hotel from the hotels I've stayed in
- Average occupancy per room – Since hotels need to be profitable, let's guess that each night a room has an 80% chance of being occupied.
- This makes the formula: 60,000 * 50 * 1 * 0.8 * 365 = 876 million bottles

NOTES

The solution to this question is to come up with a formula for the number you're trying to estimate, then take a guess at the numbers to put in the formula. There are many, many versions of this question, from "how many ping-pong balls can fit into a Boeing 747 airplane?" to "how many pianos are there in France?" The interviewer is looking to see that you can come up with a formula that makes some sense and that your logic for guessing each of the numbers in the formula makes sense. There is almost no chance you'll get the number close to right during the interview, for instance we have no idea if 50 is a good guess for the average number of hotel rooms, but that's not important.

There isn't much you can do to prepare for these questions except practicing at the improvisational component of coming up with formulas and estimates on the fly.

A.5.2 Combinatorics



"Suppose there is a grid like the one pictured, and a mouse is at the lower left corner of the grid. At the upper right corner is a piece of cheese. The mouse can only travel along the lines in the grid and would never move away from it. How many paths are there from the mouse to the cheese?"

EXAMPLE ANSWER

To get to the cheese the mouse has to move one space alone a horizontal line in the grid nine times and move one space along a vertical line in the grid six times (since the grid is 9x6). Let's call a horizontal move H and a vertical move V. Then any string with 9 Hs and Vs is a valid path from the start to the end. For example going straight up and then to the right would be VVVVVVHHHHHHHHH. There are 15 factorial (15!) ways to arrange 15 distinct characters, which are called permutations, but since 6 of them are the same letter (V) and 9 of them are the letter (H) we have to remove all the duplicate arrangements. We can remove them by counting how many duplicates there are of each, and the Vs are duplicated 6! times (the number of ways they can be arrange) while the Hs are duplicated 9! times. That means the answer is 15!/(6!)/(9!), or 5,005 paths.

NOTES

This is a really hard question to answer. First, it's hard to know the right answer—if you've somehow studied the field of combinatorics you might know it, otherwise it's hard to suddenly realize you can think of it as arranging paths. Even if you see that way of formulating the

problem, you might not know how to count the number of solutions there are to it. Second, even if you know the answer it's hard to give it in a way that clearly explains the problem and solution without being verbose. Terms like permutation are not something you can assume are known to everybody, and yet if you were to explain it all you would spend too much time on it. Lastly, there is really no way to study for it, there are so many combinatorics types question you can have answers for all of them prepared in advance. Your best bet on questions like these are to explain your thought process and how you might approach the problem. If the interview is putting a lot of weight into questions like these, that is a red flag.