CHAPTER 2

Statistical Inference, Exploratory Data Analysis, and the Data Science Process

We begin this chapter with a discussion of statistical inference and statistical thinking. Next we explore what we feel every data scientist should do once they've gotten data in hand for any data-related project: exploratory data analysis (EDA).

From there, we move into looking at what we're defining as the data science process in a little more detail. We'll end with a thought experiment and a case study.

Statistical Thinking in the Age of Big Data

> Big Data is a vague term, used loosely, if often, these days. But put simply, the catchall phrase means three things. First, it is a bundle of technologies. Second, it is a potential revolution in measurement. And third, it is a point of view, or philosophy, about how decisions will be—and perhaps should be—made in the future.
>
> — Steve Lohr
> *The New York Times*

When you're developing your skill set as a data scientist, certain foundational pieces need to be in place first—statistics, linear algebra, some programming. Even once you have those pieces, part of the challenge is that you will be developing several skill sets in parallel simultaneously—data preparation and munging, modeling, coding, visualization, and communication—that are interdependent. As we progress

through the book, these threads will be intertwined. That said, we need to start somewhere, and will begin by getting grounded in statistical inference.

We expect the readers of this book to have diverse backgrounds. For example, some of you might already be awesome software engineers who can build data pipelines and code with the best of them but don't know much about statistics; others might be marketing analysts who don't really know how to code at all yet; and others might be curious, smart people who want to know what this data science thing is all about.

So while we're asking that readers already have certain prerequisites down, we can't come to your house and look at your transcript to make sure you actually have taken a statistics course, or have read a statistics book before. And even if you have taken Introduction to Statistics—a course we know from many awkward cocktail party conversations that 99% of people dreaded and wish they'd never had to take—this likely gave you no flavor for the depth and beauty of statistical inference.

But even if it did, and maybe you're a PhD-level statistician, it's always helpful to go back to fundamentals and remind ourselves of what statistical inference and thinking is all about. And further still, in the age of Big Data, classical statistics methods need to be revisited and reimagined in new contexts.

Statistical Inference

The world we live in is complex, random, and uncertain. At the same time, it's one big data-generating machine.

As we commute to work on subways and in cars, as our blood moves through our bodies, as we're shopping, emailing, procrastinating at work by browsing the Internet and watching the stock market, as we're building things, eating things, talking to our friends and family about things, while factories are producing products, this all at least potentially produces data.

Imagine spending 24 hours looking out the window, and for every minute, counting and recording the number of people who pass by. Or gathering up everyone who lives within a mile of your house and making them tell you how many email messages they receive every day for the next year. Imagine heading over to your local hospital and rummaging around in the blood samples looking for patterns in the

DNA. That all sounded creepy, but it wasn't supposed to. The point here is that the processes in our lives are actually data-generating processes.

We'd like ways to describe, understand, and make sense of these processes, in part because as scientists we just want to understand the world better, but many times, understanding these processes is part of the solution to problems we're trying to solve.

Data represents the traces of the real-world processes, and exactly which traces we gather are decided by our data collection or sampling method. You, the data scientist, the observer, are turning the world into data, and this is an utterly subjective, not objective, process.

After separating the process from the data collection, we can see clearly that there are two sources of randomness and uncertainty. Namely, the randomness and uncertainty underlying the process itself, and the uncertainty associated with your underlying data collection methods.

Once you have all this data, you have somehow captured the world, or certain traces of the world. But you can't go walking around with a huge Excel spreadsheet or database of millions of transactions and look at it and, with a snap of a finger, understand the world and process that generated it.

So you need a new idea, and that's to simplify those captured traces into something more comprehensible, to something that somehow captures it all in a much more concise way, and that something could be mathematical models or functions of the data, known as statistical estimators.

This overall process of going from the world to the data, and then from the data back to the world, is the field of *statistical inference*.

More precisely, statistical inference is the discipline that concerns itself with the development of procedures, methods, and theorems that allow us to extract meaning and information from data that has been generated by stochastic (random) processes.

Populations and Samples

Let's get some terminology and concepts in place to make sure we're all talking about the same thing.

In classical statistical literature, a distinction is made between the population and the sample. The word *population* immediately makes

us think of the entire US population of 300 million people, or the entire world's population of 7 billion people. But put that image out of your head, because in statistical inference population isn't used to simply describe only people. It could be any set of objects or units, such as tweets or photographs or stars.

If we could measure the characteristics or extract characteristics of all those objects, we'd have a complete set of *observations*, and the convention is to use N to represent the total number of observations in the population.

Suppose your population was all emails sent last year by employees at a huge corporation, BigCorp. Then a single observation could be a list of things: the sender's name, the list of recipients, date sent, text of email, number of characters in the email, number of sentences in the email, number of verbs in the email, and the length of time until first reply.

When we take a *sample*, we take a subset of the units of size n in order to examine the observations to draw conclusions and make inferences about the population. There are different ways you might go about getting this subset of data, and you want to be aware of this sampling mechanism because it can introduce *biases* into the data, and distort it, so that the subset is not a "mini-me" shrunk-down version of the population. Once that happens, any conclusions you draw will simply be wrong and distorted.

In the BigCorp email example, you could make a list of all the employees and select 1/10th of those people *at random* and take all the email they ever sent, and that would be your sample. Alternatively, you could sample 1/10th of all email sent each day at random, and that would be your sample. Both these methods are reasonable, and both methods yield the same sample size. But if you took them and counted how many email messages each person sent, and used that to estimate the underlying *distribution* of emails sent by all indiviuals at BigCorp, you might get entirely different answers.

So if even getting a basic thing down like counting can get distorted when you're using a reasonable-sounding sampling method, imagine what can happen to more complicated algorithms and models if you haven't taken into account the process that got the data into your hands.

Populations and Samples of Big Data

But, wait! In the age of Big Data, where we can record all users' actions all the time, don't we observe *everything*? Is there really still this notion of population and sample? If we had all the email in the first place, why would we need to take a sample?

With these questions, we've gotten to the heart of the matter. There are multiple aspects of this that need to be addressed.

Sampling solves some engineering challenges
: In the current popular discussion of Big Data, the focus on enterprise solutions such as Hadoop to handle engineering and computational challenges caused by too much data overlooks sampling as a legitimate solution. At Google, for example, software engineers, data scientists, and statisticians sample all the time.

How much data you need at hand really depends on what your goal is: for analysis or inference purposes, you typically don't need to store all the data all the time. On the other hand, for serving purposes you might: in order to render the correct information in a UI for a user, you need to have all the information for that particular user, for example.

Bias
: Even if we have access to all of Facebook's or Google's or Twitter's data corpus, any inferences we make from that data should not be extended to draw conclusions about humans beyond those sets of users, or even those users for any particular day.

Kate Crawford, a principal scientist at Microsoft Research, describes in her Strata talk, "Hidden Biases of Big Data," how if you analyzed tweets immediately before and after Hurricane Sandy, you would think that most people were supermarket shopping pre-Sandy and partying post-Sandy. However, most of those tweets came from New Yorkers. First of all, they're heavier Twitter users than, say, the coastal New Jerseyans, and second of all, the coastal New Jerseyans were worrying about other stuff like their house falling down and didn't have time to tweet.

In other words, you would think that Hurricane Sandy wasn't all that bad if you used tweet data to understand it. The only conclusion you can actually draw is that this is what Hurricane Sandy was like for the subset of Twitter users (who themselves are not representative of the

general US population), whose situation was not so bad that they didn't have time to tweet.

Note, too, that in this case, if you didn't *have context* and know about Hurricane Sandy, you wouldn't know enough to interpret this data properly.

Sampling

Let's rethink what the population and the sample are in various contexts.

In statistics we often model the relationship between a population and a sample with an underlying mathematical process. So we make simplifying *assumptions* about the underlying truth, the mathematical structure, and shape of the underlying generative process that created the data. We observe only one particular realization of that generative process, which is that sample.

So if we think of all the emails at BigCorp as the population, and if we randomly sample from that population by reading some but not all emails, then that sampling process would create one particular sample. However, if we resampled we'd get a different set of observations.

The uncertainty created by such a sampling process has a name: the *sampling distribution*. But like that 2010 movie *Inception* with Leonardo DiCaprio, where he's in a dream within a dream within a dream, it's possible to instead think of the complete corpus of emails at BigCorp as not the population but as a sample.

This set of emails (and here is where we're getting philosophical, but that's what this is all about) could actually be only one single realization from some larger *super-population*, and if the Great Coin Tosser in the sky had spun again that day, a different set of emails would have been observed.

In this interpretation, we treat this set of emails as a sample that we are using to make inferences about the underlying generative process that is the email writing habits of all the employees at BigCorp.

New kinds of data

Gone are the days when data is just a bunch of numbers and categorical variables. A strong data scientist needs to be versatile and comfortable with dealing a variety of types of data, including:

- Traditional: numerical, categorical, or binary
- Text: emails, tweets, *New York Times* articles (see Chapter 4 or Chapter 7)
- Records: user-level data, timestamped event data, json-formatted log files (see Chapter 6 or Chapter 8)
- Geo-based location data: briefly touched on in this chapter with NYC housing data
- Network (see Chapter 10)
- Sensor data (not covered in this book)
- Images (not covered in this book)

These new kinds of data require us to think more carefully about what sampling means in these contexts.

For example, with the firehose of real-time streaming data, if you analyze a Facebook user-level dataset for a week of activity that you aggregated from timestamped event logs, will any conclusions you draw from this dataset be relevant next week or next year?

How do you sample from a network and preserve the complex network structure?

Many of these questions represent open research questions for the statistical and computer science communities. This is the frontier! Given that some of these are open research problems, in practice, data scientists do the best they can, and often are inventing novel methods as part of their jobs.

Terminology: Big Data

We've been throwing around "Big Data" quite a lot already and are guilty of barely defining it beyond raising some big questions in the previous chapter.

A few ways to think about Big Data:

"Big" is a moving target. Constructing a threshold for Big Data such as 1 petabyte is meaningless because it makes it sound absolute. Only when the size becomes a challenge is it worth referring to it as "Big." So it's a relative term referring to when the size of the data outstrips the state-of-the-art current computational solutions (in terms of memory, storage, complexity, and processing speed) available to handle it. So in the 1970s this meant something different than it does today.

"Big" is when you can't fit it on one machine. Different individuals and companies have different computational resources available to them, so for a single scientist data is big if she can't fit it on one machine because she has to learn a whole new host of tools and methods once that happens.

Big Data is a cultural phenomenon. It describes how much data is part of our lives, precipitated by accelerated advances in technology.

The 4 Vs: Volume, variety, velocity, and value. Many people are circulating this as a way to characterize Big Data. Take from it what you will.

Big Data Can Mean Big Assumptions

In Chapter 1, we mentioned the Cukier and Mayer-Schoenberger article "The Rise of Big Data." In it, they argue that the Big Data revolution consists of three things:

- Collecting and using a lot of data rather than small samples
- Accepting messiness in your data
- Giving up on knowing the causes

They describe these steps in a rather grand fashion by claiming that Big Data doesn't need to understand cause given that the data is so enormous. It doesn't need to worry about sampling error because it is

literally *keeping track of the truth*. The way the article frames this is by claiming that the new approach of Big Data is letting "N=ALL."

Can N=ALL?

Here's the thing: it's pretty much never all. And we are very often missing the very things we should care about most.

So, for example, as this InfoWorld post explains, Internet surveillance will never really work, because the very clever and tech-savvy criminals that we most want to catch are the very ones we will never be able to catch, because they're always a step ahead.

An example from that very article—election night polls—is in itself a great counter-example: even if we poll absolutely everyone who leaves the polling stations, we still don't count people who decided not to vote in the first place. And those might be the very people we'd need to talk to to understand our country's voting problems.

Indeed, we'd argue that the assumption we make that N=ALL is one of the biggest problems we face in the age of Big Data. It is, above all, a way of excluding the voices of people who don't have the time, energy, or access to cast their vote in all sorts of informal, possibly unannounced, elections.

Those people, busy working two jobs and spending time waiting for buses, become invisible when we tally up the votes without them. To you this might just mean that the recommendations you receive on Netflix don't seem very good because most of the people who bother to rate things on Netflix are young and might have different tastes than you, which skews the recommendation engine toward them. But there are plenty of much more insidious consequences stemming from this basic idea.

Data is not objective

Another way in which the assumption that N=ALL can matter is that it often gets translated into the idea that data is *objective*. It is wrong to believe either that data is objective or that "data speaks," and beware of people who say otherwise.

We were recently reminded of it in a terrifying way by this New York Times article on Big Data and recruiter hiring practices. At one point, a data scientist is quoted as saying, "Let's put everything in and let the data speak for itself."

If you read the whole article, you'll learn that this algorithm tries to find "diamond in the rough" types of people to hire. A worthy effort, but one that you have to think through.

Say you decided to compare women and men with the exact same qualifications that have been hired in the past, but then, looking into what happened next you learn that those women have tended to leave more often, get promoted less often, and give more negative feedback on their environments when compared to the men.

Your model might be likely to hire the man over the woman next time the two similar candidates showed up, rather than looking into the possibility that the company doesn't treat female employees well.

In other words, ignoring causation can be a flaw, rather than a feature. Models that ignore causation can add to historical problems instead of addressing them (we'll explore this more in Chapter 11). And data doesn't speak for itself. Data is just a quantitative, pale echo of the events of our society.

n = 1

At the other end of the spectrum from N=ALL, we have $n = 1$, by which we mean a sample size of 1. In the old days a sample size of 1 would be ridiculous; you would never want to draw inferences about an entire population by looking at a single individual. And don't worry, that's still ridiculous. But the concept of $n = 1$ takes on new meaning in the age of Big Data, where for a single person, we actually can record tons of information about them, and in fact we might even sample from all the events or actions they took (for example, phone calls or keystrokes) in order to make inferences about them. This is what user-level modeling is about.

Modeling

In the next chapter, we'll look at how we build models from the data we collect, but first we want to discuss what we even mean by this term.

Rachel had a recent phone conversation with someone about a *modeling* workshop, and several minutes into it she realized the word "model" meant completely different things to them. He was using it to mean *data models*—the representation one is choosing to store one's data, which is the realm of database managers—whereas she was

talking about *statistical models*, which is what much of this book is about. One of Andrew Gelman's blog posts on modeling was recently tweeted by people in the fashion industry, but that's a different issue.

Even if you've used the terms *statistical model* or *mathematical model* for years, is it even clear to yourself and to the people you're talking to what you mean? What makes a model a *model*? Also, while we're asking fundamental questions like this, what's the difference between a statistical model and a machine learning algorithm?

Before we dive deeply into that, let's add a bit of context with this deliberately provocative *Wired* magazine piece, "The End of Theory: The Data Deluge Makes the Scientific Method Obsolete," published in 2008 by Chris Anderson, then editor-in-chief.

Anderson equates massive amounts of data to complete information and argues no models are necessary and "correlation is enough"; e.g., that in the context of massive amounts of data, "they [Google] don't have to settle for models at all."

Really? We don't think so, and we don't think you'll think so either by the end of the book. But the sentiment is similar to the Cukier and Mayer-Schoenberger article we just discussed about N=ALL, so you might already be getting a sense of the profound confusion we're witnessing all around us.

To their credit, it's the press that's currently raising awareness of these questions and issues, and someone has to do it. Even so, it's hard to take when the opinion makers are people who don't actually work with data. Think critically about whether you buy what Anderson is saying; where you agree, disagree, or where you need more information to form an opinion.

Given that this is how the popular press is currently describing and influencing public perception of data science and modeling, it's incumbent upon us as data scientists to be aware of it and to chime in with informed comments.

With that context, then, what do we mean when we say *models*? And how do we use them as data scientists? To get at these questions, let's dive in.

What is a model?

Humans try to understand the world around them by representing it in different ways. Architects capture attributes of buildings through

blueprints and three-dimensional, scaled-down versions. Molecular biologists capture protein structure with three-dimensional visualizations of the connections between amino acids. Statisticians and data scientists capture the uncertainty and randomness of data-generating processes with mathematical functions that express the shape and structure of the data itself.

A model is our attempt to understand and represent the nature of reality through a particular lens, be it architectural, biological, or mathematical.

A model is an artificial construction where all extraneous detail has been removed or abstracted. Attention must always be paid to these abstracted details after a model has been analyzed to see what might have been overlooked.

In the case of proteins, a model of the protein backbone with side-chains by itself is removed from the laws of quantum mechanics that govern the behavior of the electrons, which ultimately dictate the structure and actions of proteins. In the case of a statistical model, we may have mistakenly excluded key variables, included irrelevant ones, or assumed a mathematical structure divorced from reality.

Statistical modeling

Before you get too involved with the data and start coding, it's useful to draw a picture of what you think the underlying process might be with your model. What comes first? What influences what? What causes what? What's a test of that?

But different people think in different ways. Some prefer to express these kinds of relationships in terms of math. The mathematical expressions will be general enough that they have to include parameters, but the values of these parameters are not yet known.

In mathematical expressions, the convention is to use Greek letters for parameters and Latin letters for data. So, for example, if you have two columns of data, x and y, and you think there's a linear relationship, you'd write down $y = \beta_0 + \beta_1 x$. You don't know what β_0 and β_1 are in terms of actual numbers yet, so they're the parameters.

Other people prefer pictures and will first draw a diagram of data flow, possibly with arrows, showing how things affect other things or what happens over time. This gives them an abstract picture of the relationships before choosing equations to express them.

But how do you build a model?

How do you have any clue whatsoever what functional form the data should take? Truth is, it's part art and part science. And sadly, this is where you'll find the least guidance in textbooks, in spite of the fact that it's the key to the whole thing. After all, this is the part of the modeling process where you have to make a lot of assumptions about the underlying structure of reality, and we should have standards as to how we make those choices and how we explain them. But we don't have global standards, so we make them up as we go along, and hopefully in a thoughtful way.

We're admitting this here: where to start is not obvious. If it were, we'd know the meaning of life. However, we will do our best to demonstrate for you throughout the book how it's done.

One place to start is exploratory data analysis (EDA), which we will cover in a later section. This entails making plots and building intuition for your particular dataset. EDA helps out a lot, as well as trial and error and iteration.

To be honest, until you've done it a lot, it seems very mysterious. The best thing to do is start simply and then build in complexity. Do the dumbest thing you can think of first. It's probably not that dumb.

For example, you can (and should) plot histograms and look at scatterplots to start getting a feel for the data. Then you just try writing something down, even if it's wrong first (it will probably be wrong first, but that doesn't matter).

So try writing down a linear function (more on that in the next chapter). When you write it down, you force yourself to think: does this make *any* sense? If not, why? What would make *more sense*? You start simply and keep building it up in complexity, making assumptions, and writing your assumptions down. You can use full-blown sentences if it helps—e.g., "I assume that my users naturally cluster into about five groups because when I hear the sales rep talk about them, she has about five different types of people she talks about"—then taking your words and trying to express them as equations and code.

Remember, it's always good to start simply. There is a trade-off in modeling between simple and accurate. Simple models may be easier to interpret and understand. Oftentimes the crude, simple model gets you 90% of the way there and only takes a few hours to build and fit,

whereas getting a more complex model might take months and only get you to 92%.

You'll start building up your arsenal of potential models throughout this book. Some of the building blocks of these models are *probability distributions*.

Probability distributions

Probability distributions are the foundation of statistical models. When we get to linear regression and Naive Bayes, you will see how this happens in practice. One can take multiple semesters of courses on probability theory, and so it's a tall challenge to condense it down for you in a small section.

Back in the day, before computers, scientists observed real-world phenomenon, took measurements, and noticed that certain mathematical shapes kept reappearing. The classical example is the height of humans, following a *normal* distribution—a bell-shaped curve, also called a Gaussian distribution, named after Gauss.

Other common shapes have been named after their observers as well (e.g., the Poisson distribution and the Weibull distribution), while other shapes such as Gamma distributions or exponential distributions are named after associated mathematical objects.

Natural processes tend to generate measurements whose empirical shape could be approximated by mathematical functions with a few parameters that could be estimated from the data.

Not *all* processes generate data that looks like a *named* distribution, but many do. We can use these functions as building blocks of our models. It's beyond the scope of the book to go into each of the distributions in detail, but we provide them in Figure 2-1 as an illustration of the various common shapes, and to remind you that they only have names because someone observed them enough times to think they deserved names. There is actually an infinite number of possible distributions.

They are to be interpreted as assigning a *probability* to a subset of possible outcomes, and have corresponding functions. For example, the normal distribution is written as:

$$N(x|\mu,\sigma) \sim \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

The parameter μ is the mean and median and controls where the distribution is centered (because this is a symmetric distribution), and the parameter σ controls how spread out the distribution is. This is the general functional form, but for specific real-world phenomenon, these parameters have actual numbers as values, which we can estimate from the data.

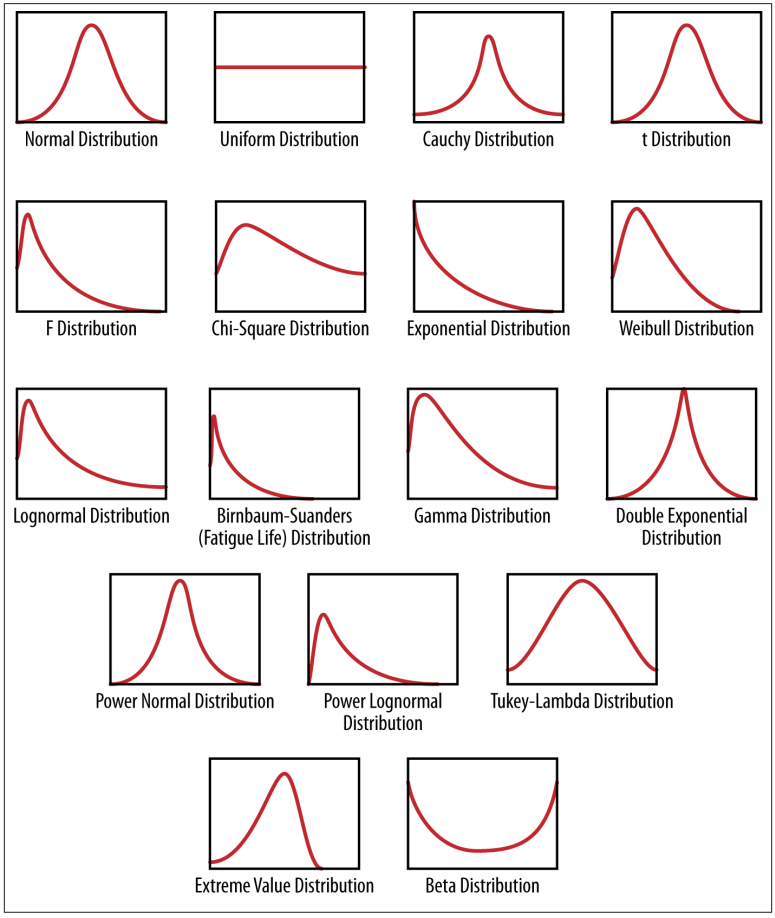

Figure 2-1. A bunch of continuous density functions (aka probability distributions)

A *random variable* denoted by x or y can be assumed to have a corresponding probability distribution, $p(x)$, which maps x to a positive real number. In order to be a probability density function, we're restricted to the set of functions such that if we integrate $p(x)$ to get the area under the curve, it is 1, so it can be interpreted as probability.

For example, let x be the amount of time until the next bus arrives (measured in seconds). x is a random variable because there is variation and uncertainty in the amount of time until the next bus.

Suppose we know (for the sake of argument) that the time until the next bus has a probability density function of $p(x) = 2e^{-2x}$. If we want to know the likelihood of the next bus arriving in between 12 and 13 minutes, then we find the area under the curve between 12 and 13 by $\int_{12}^{13} 2e^{-2x}$.

How do we know this is the right distribution to use? Well, there are two possible ways: we can conduct an experiment where we show up at the bus stop at a random time, measure how much time until the next bus, and repeat this experiment over and over again. Then we look at the measurements, plot them, and approximate the function as discussed. Or, because we are familiar with the fact that "waiting time" is a common enough real-world phenomenon that a distribution called the exponential distribution has been invented to describe it, we know that it takes the form $p(x) = \lambda e^{-\lambda x}$.

In addition to denoting distributions of single random variables with functions of one variable, we use multivariate functions called *joint distributions* to do the same thing for more than one random variable. So in the case of two random variables, for example, we could denote our distribution by a function $p(x, y)$, and it would take values in the plane and give us nonnegative values. In keeping with its interpretation as a probability, its (double) integral over the whole plane would be 1.

We also have what is called a *conditional distribution*, $p(x|y)$, which is to be interpreted as the density function of x given a particular value of y.

When we're working with data, conditioning corresponds to subsetting. So for example, suppose we have a set of user-level data for Amazon.com that lists for each user the amount of money spent last month on Amazon, whether the user is male or female, and how many items they looked at before adding the first item to the shopping cart.

If we consider X to be the random variable that represents the amount of money spent, then we can look at the distribution of money spent across all users, and represent it as $p(X)$.

We can then take the subset of users who looked at more than five items before buying anything, and look at the distribution of money spent among these users. Let Y be the random variable that represents number of items looked at, then $p(X \mid Y > 5)$ would be the corresponding *conditional distribution*. Note a conditional distribution has the same properties as a regular distribution in that when we integrate it, it sums to 1 and has to take nonnegative values.

When we observe data points, i.e., $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, we are observing *realizations* of a pair of random variables. When we have an entire dataset with n rows and k columns, we are observing n realizations of the joint distribution of those k random variables.

For further reading on probability distributions, we recommend Sheldon Ross' book, *A First Course in Probability* (Pearson).

Fitting a model

Fitting a model means that you estimate the parameters of the model using the observed data. You are using your data as evidence to help approximate the real-world mathematical process that generated the data. Fitting the model often involves optimization methods and algorithms, such as *maximum likelihood estimation*, to help get the parameters.

In fact, when you estimate the parameters, they are actually *estimators*, meaning they themselves are *functions* of the data. Once you fit the model, you actually can write it as $y = 7.2 + 4.5x$, for example, which means that your best guess is that this equation or functional form expresses the relationship between your two variables, based on your assumption that the data followed a linear pattern.

Fitting the model is when you start actually coding: your code will read in the data, and you'll specify the functional form that you wrote down on the piece of paper. Then R or Python will use built-in optimization methods to give you the most likely values of the parameters given the data.

As you gain sophistication, or if this is one of your areas of expertise, you'll dig around in the optimization methods yourself. Initially you should have an understanding that optimization is taking place and

how it works, but you don't have to code this part yourself—it underlies the R or Python functions.

Overfitting

Throughout the book you will be cautioned repeatedly about *overfitting*, possibly to the point you will have nightmares about it. Overfitting is the term used to mean that you used a dataset to estimate the parameters of your model, but your model isn't that good at capturing reality beyond your sampled data.

You might know this because you have tried to use it to predict labels for another set of data that you didn't use to fit the model, and it doesn't do a good job, as measured by an evaluation metric such as accuracy.

Exploratory Data Analysis

> "Exploratory data analysis" is an attitude, a state of flexibility, a willingness to look for those things that we believe are not there, as well as those we believe to be there.
>
> — John Tukey

Earlier we mentioned exploratory data analysis (EDA) as the first step toward building a model. EDA is often relegated to chapter 1 (by which we mean the "easiest" and lowest level) of standard introductory statistics textbooks and then forgotten about for the rest of the book.

It's traditionally presented as a bunch of histograms and stem-and-leaf plots. They teach that stuff to kids in fifth grade so it seems trivial, right? No wonder no one thinks much of it.

But EDA is a critical part of the data science process, and also represents a philosophy or way of doing statistics practiced by a strain of statisticians coming from the Bell Labs tradition.

John Tukey, a mathematician at Bell Labs, developed exploratory data analysis in contrast to confirmatory data analysis, which concerns itself with modeling and hypotheses as described in the previous section. In EDA, there is no hypothesis and there is no model. The "exploratory" aspect means that your understanding of the problem you are solving, or might solve, is changing as you go.

Historical Perspective: Bell Labs

Bell Labs is a research lab going back to the 1920s that has made innovations in physics, computer science, statistics, and math, producing languages like C++, and many Nobel Prize winners as well. There was a very successful and productive statistics group there, and among its many notable members was John Tukey, a mathematician who worked on a lot of statistical problems. He is considered the father of EDA and R (which started as the S language at Bell Labs; R is the open source version), and he was interested in trying to visualize high-dimensional data.

We think of Bell Labs as one of the places where data science was "born" because of the collaboration between disciplines, and the massive amounts of complex data available to people working there. It was a virtual playground for statisticians and computer scientists, much like Google is today.

In fact, in 2001, Bill Cleveland wrote "Data Science: An Action Plan for expanding the technical areas of the field of statistics," which described multidisciplinary investigation, models, and methods for data (traditional applied stats), computing with data (hardware, software, algorithms, coding), pedagogy, tool evaluation (staying on top of current trends in technology), and theory (the math behind the data).

You can read more about Bell Labs in the book *The Idea Factory* by Jon Gertner (Penguin Books).

The basic tools of EDA are plots, graphs and summary statistics. Generally speaking, it's a method of systematically going through the data, plotting distributions of all variables (using box plots), plotting time series of data, transforming variables, looking at all pairwise relationships between variables using scatterplot matrices, and generating summary statistics for all of them. At the very least that would mean computing their mean, minimum, maximum, the upper and lower quartiles, and identifying outliers.

But as much as EDA is a set of tools, it's also a mindset. And that mindset is about your relationship with the data. You want to understand the data—gain intuition, understand the shape of it, and try to connect your understanding of the process that generated the data to

the data itself. EDA happens between you and the data and isn't about proving anything to anyone else yet.

Philosophy of Exploratory Data Analysis

> Long before worrying about how to convince others, you first have to understand what's happening yourself.
>
> — Andrew Gelman

While at Google, Rachel was fortunate to work alongside two former Bell Labs/AT&T statisticians—Daryl Pregibon and Diane Lambert, who also work in this vein of applied statistics—and learned from them to make EDA a part of her best practices.

Yes, even with very large Google-scale data, they did EDA. In the context of data in an Internet/engineering company, EDA is done for some of the same reasons it's done with smaller datasets, but there are additional reasons to do it with data that has been generated from logs.

There are important reasons anyone working with data should do EDA. Namely, to gain intuition about the data; to make comparisons between distributions; for sanity checking (making sure the data is on the scale you expect, in the format you thought it should be); to find out where data is missing or if there are outliers; and to summarize the data.

In the context of data generated from logs, EDA also helps with debugging the logging process. For example, "patterns" you find in the data could actually be something wrong in the logging process that needs to be fixed. If you never go to the trouble of debugging, you'll continue to think your patterns are real. The engineers we've worked with are always grateful for help in this area.

In the end, EDA helps you make sure the product is performing as intended.

Although there's lots of visualization involved in EDA, we distinguish between EDA and data visualization in that EDA is done toward the beginning of analysis, and data visualization (which we'll get to in Chapter 9), as it's used in our vernacular, is done toward the end to communicate one's findings. With EDA, the graphics are solely done for *you* to understand what's going on.

With EDA, you can also use the understanding you get to inform and improve the development of algorithms. For example, suppose you

are trying to develop a ranking algorithm that ranks content that you are showing to users. To do this you might want to develop a notion of "popular."

Before you decide how to quantify popularity (which could be, for example, highest frequency of clicks, or the post with the most number of comments, or comments above some threshold, or some weighted average of many metrics), you need to understand how the data is behaving, and the best way to do that is looking at it and getting your hands dirty.

Plotting data and making comparisons can get you extremely far, and is far better to do than getting a dataset and immediately running a regression just because you know how. It's been a disservice to analysts and data scientists that EDA has not been enforced as a critical part of the process of working with data. Take this opportunity to make it part of your process!

Here are some references to help you understand best practices and historical context:

1. *Exploratory Data Analysis* by John Tukey (Pearson)
2. *The Visual Display of Quantitative Information* by Edward Tufte (Graphics Press)
3. *The Elements of Graphing Data* by William S. Cleveland (Hobart Press)
4. *Statistical Graphics for Visualizing Multivariate Data* by William G. Jacoby (Sage)
5. "Exploratory Data Analysis for Complex Models" by Andrew Gelman (American Statistical Association)
6. *The Future of Data Analysis* by John Tukey. Annals of Mathematical Statistics, Volume 33, Number 1 (1962), 1-67.
7. *Data Analysis, Exploratory* by David Brillinger [8-page excerpt from *International Encyclopedia of Political Science* (Sage)]

Exercise: EDA

There are 31 datasets named nyt1.csv, nyt2.csv,…,nyt31.csv, which you can find here: *https://github.com/oreillymedia/doing_data_science*.

Each one represents one (simulated) day's worth of ads shown and clicks recorded on the *New York Times* home page in May 2012. Each row represents a single user. There are five columns: age, gender (0=female, 1=male), number impressions, number clicks, and logged-in.

You'll be using R to handle these data. It's a programming language designed specifically for data analysis, and it's pretty intuitive to start using. You can download it here. Once you have it installed, you can load a single file into R with this command:

```
data1 <- read.csv(url("http://stat.columbia.edu/~rachel/
                      datasets/nyt1.csv"))
```

Once you have the data loaded, it's time for some EDA:

1. Create a new variable, `age_group`, that categorizes users as `"<18"`, `"18-24"`, `"25-34"`, `"35-44"`, `"45-54"`, `"55-64"`, and `"65+"`.
2. For a single day:
   - Plot the distributions of number impressions and click-through-rate (CTR=# clicks/# impressions) for these six age categories.
   - Define a new variable to segment or categorize users based on their click behavior.
   - Explore the data and make visual and quantitative comparisons across user segments/demographics (<18-year-old males versus < 18-year-old females or logged-in versus not, for example).
   - Create metrics/measurements/statistics that summarize the data. Examples of potential metrics include CTR, quantiles, mean, median, variance, and max, and these can be calculated across the various user segments. Be selective. Think about what will be important to track over time—what will compress the data, but still capture user behavior.
3. Now extend your analysis across days. Visualize some metrics and distributions over time.
4. Describe and interpret any patterns you find.

Sample code

Here we'll give you the beginning of a sample solution for this exercise. The reality is that we can't teach you about data science and teach you

how to code all in the same book. Learning to code in a new language requires a lot of trial and error as well as going online and searching on Google or stackoverflow.

Chances are, if you're trying to figure out how to plot something or build a model in R, other people have tried as well, so rather than banging your head against the wall, look online. [Ed note: There might also be some books available to help you out on this front as well.] We suggest not looking at this code until you've struggled along a bit:

```
# Author: Maura Fitzgerald
data1 <- read.csv(url("http://stat.columbia.edu/~rachel/
                      datasets/nyt1.csv"))

# categorize
head(data1)
data1$agecat <-cut(data1$Age,c(-Inf,0,18,24,34,44,54,64,Inf))

# view
summary(data1)

# brackets
install.packages("doBy")
library("doBy")
siterange <- function(x){c(length(x), min(x), mean(x), max(x))}
summaryBy(Age~agecat, data =data1, FUN=siterange)

# so only signed in users have ages and genders
summaryBy(Gender+Signed_In+Impressions+Clicks~agecat,
          data =data1)

# plot
install.packages("ggplot2")
library(ggplot2)
ggplot(data1, aes(x=Impressions, fill=agecat))
       +geom_histogram(binwidth=1)
ggplot(data1, aes(x=agecat, y=Impressions, fill=agecat))
       +geom_boxplot()

# create click thru rate
# we don't care about clicks if there are no impressions
# if there are clicks with no imps my assumptions about
# this data are wrong
data1$hasimps <-cut(data1$Impressions,c(-Inf,0,Inf))
summaryBy(Clicks~hasimps, data =data1, FUN=siterange)
ggplot(subset(data1, Impressions>0), aes(x=Clicks/Impressions,
       colour=agecat)) + geom_density()
ggplot(subset(data1, Clicks>0), aes(x=Clicks/Impressions,
       colour=agecat)) + geom_density()
ggplot(subset(data1, Clicks>0), aes(x=agecat, y=Clicks,
```

```
      fill=agecat)) + geom_boxplot()
ggplot(subset(data1, Clicks>0), aes(x=Clicks, colour=agecat))
      + geom_density()

# create categories
data1$scode[data1$Impressions==0] <- "NoImps"
data1$scode[data1$Impressions >0] <- "Imps"
data1$scode[data1$Clicks >0] <- "Clicks"

# Convert the column to a factor
data1$scode <- factor(data1$scode)
head(data1)

#look at levels
clen <- function(x){c(length(x))}
etable<-summaryBy(Impressions~scode+Gender+agecat,
                  data = data1, FUN=clen)
```

Hint for doing the rest: don't read all the datasets into memory. Once you've perfected your code for one day, read the datasets in one at a time, process them, output any relevant metrics and variables, and store them in a dataframe; then remove the dataset before reading in the next one. This is to get you thinking about how to handle data sharded across multiple machines.

On Coding

In a May 2013 op-ed piece, "How to be a Woman Programmer," Ellen Ullman describes quite well what it takes to be a programmer (setting aside for now the woman part):

"The first requirement for programming is a passion for the work, a deep need to probe the mysterious space between human thoughts and what a machine can understand; between human desires and how machines might satisfy them.

The second requirement is a high tolerance for failure. Programming is the art of algorithm design and the craft of debugging errant code. In the words of the great John Backus, inventor of the Fortran programming language: *You need the willingness to fail all the time. You have to generate many ideas and then you have to work very hard only to discover that they don't work. And you keep doing that over and over until you find one that does work.*"

The Data Science Process

Let's put it all together into what we define as the data science process. The more examples you see of people doing data science, the more you'll find that they fit into the general framework shown in Figure 2-2. As we go through the book, we'll revisit stages of this process and examples of it in different ways.

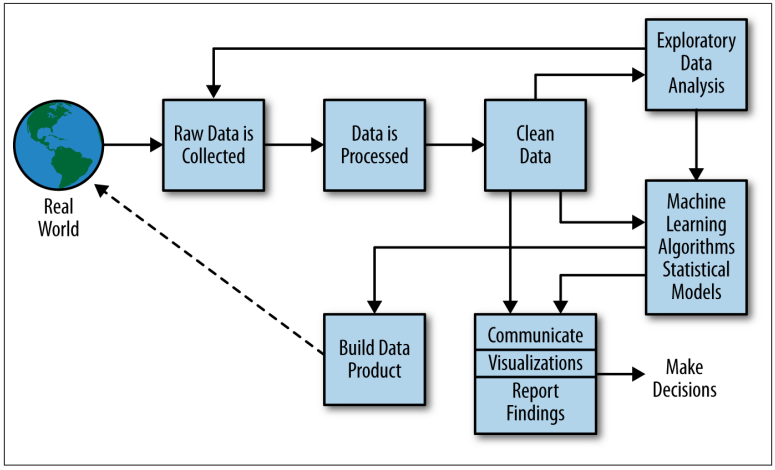

Figure 2-2. The data science process

First we have the Real World. Inside the Real World are lots of people busy at various activities. Some people are using Google+, others are competing in the Olympics; there are spammers sending spam, and there are people getting their blood drawn. Say we have data on one of these things.

Specifically, we'll start with raw data—logs, Olympics records, Enron employee emails, or recorded genetic material (note there are lots of aspects to these activities already lost even when we have that raw data). We want to process this to make it clean for analysis. So we build and use pipelines of data munging: joining, scraping, wrangling, or whatever you want to call it. To do this we use tools such as Python, shell scripts, R, or SQL, or all of the above.

Eventually we get the data down to a nice format, like something with columns:

name | event | year | gender | event time

This is where you typically *start* in a standard statistics class, with a clean, orderly dataset. But it's not where you typically start in the real world.

Once we have this clean dataset, we should be doing some kind of EDA. In the course of doing EDA, we may realize that it isn't actually clean because of duplicates, missing values, absurd outliers, and data that wasn't actually logged or incorrectly logged. If that's the case, we may have to go back to collect more data, or spend more time cleaning the dataset.

Next, we design our model to use some algorithm like k-nearest neighbor (k-NN), linear regression, Naive Bayes, or something else. The model we choose depends on the type of problem we're trying to solve, of course, which could be a classification problem, a prediction problem, or a basic description problem.

We then can interpret, visualize, report, or communicate our results. This could take the form of reporting the results up to our boss or coworkers, or publishing a paper in a journal and going out and giving academic talks about it.

Alternatively, our goal may be to build or prototype a "data product"; e.g., a spam classifier, or a search ranking algorithm, or a recommendation system. Now the key here that makes data science special and distinct from statistics is that this data product then *gets incorporated back* into the real world, and users interact with that product, and that generates more data, which creates a feedback loop.

This is very different from predicting the weather, say, where your model doesn't influence the outcome at all. For example, you might predict it will rain next week, and unless you have some powers we don't know about, you're not going to *cause* it to rain. But if you instead build a recommendation system that generates evidence that "lots of people love this book," say, then you will know that you caused that feedback loop.

Take this loop into account in any analysis you do by adjusting for any biases your model caused. Your models are not just predicting the future, but *causing* it!

A data product that is productionized and that users interact with is at one extreme and the weather is at the other, but regardless of the

type of data you work with and the "data product" that gets built on top of it—be it public policy determined by a statistical model, health insurance, or election polls that get widely reported and perhaps influence viewer opinions—you should consider the extent to which your model is influencing the very phenomenon that you are trying to observe and understand.

A Data Scientist's Role in This Process

This model so far seems to suggest this will all magically happen without human intervention. By "human" here, we mean "data scientist." Someone has to make the decisions about what data to collect, and why. That person needs to be formulating questions and hypotheses and making a plan for how the problem will be attacked. And that someone is the data scientist or our beloved data science team.

Let's revise or at least add an overlay to make clear that the data scientist needs to be involved in this process throughout, meaning they are involved in the actual coding as well as in the higher-level process, as shown in Figure 2-3.

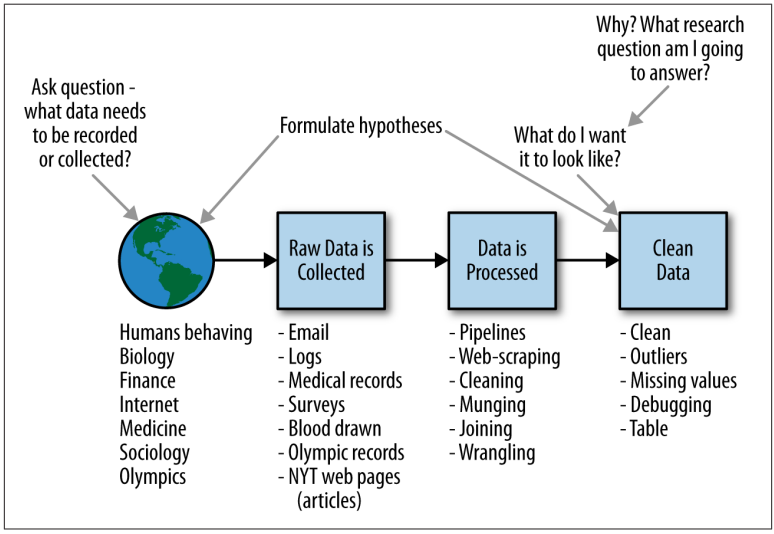

Figure 2-3. The data scientist is involved in every part of this process

Connection to the Scientific Method

We can think of the data science process as an extension of or variation of the scientific method:

- Ask a question.
- Do background research.
- Construct a hypothesis.
- Test your hypothesis by doing an experiment.
- Analyze your data and draw a conclusion.
- Communicate your results.

In both the data science process and the scientific method, not every problem requires one to go through all the steps, but almost all problems can be solved with some combination of the stages. For example, if your end goal is a data visualization (which itself could be thought of as a data product), it's possible you might not do any machine learning or statistical modeling, but you'd want to get all the way to a clean dataset, do some exploratory analysis, and then create the visualization.

Thought Experiment: How Would You Simulate Chaos?

Most data problems start out with a certain amount of dirty data, ill-defined questions, and urgency. As data scientists we are, in a sense, attempting to create order from chaos. The class took a break from the lecture to discuss how they'd simulate chaos. Here are some ideas from the discussion:

- A Lorenzian water wheel, which is a Ferris wheel-type contraption with equally spaced buckets of water that rotate around in a circle. Now imagine water being dripped into the system at the very top. Each bucket has a leak, so some water escapes into whatever bucket is directly below the drip. Depending on the rate of the water coming in, this system exhibits a chaotic process that depends on molecular-level interactions of water molecules on the sides of the buckets. Read more about it in this associated Wikipedia article.

- Many systems can exhibit inherent chaos. Philippe M. Binder and Roderick V. Jensen have written a paper entitled "Simulating chaotic behavior with finite-state machines", which is about digital computer simulations of chaos.
- An interdisciplinary program involving M.I.T., Harvard, and Tufts involved teaching a technique that was entitled "Simulating chaos to teach order". They simulated an emergency on the border between Chad and Sudan's troubled Darfur region, with students acting as members of Doctors Without Borders, International Medical Corps, and other humanitarian agencies.
- See also Joel Gascoigne's related essay, "Creating order from chaos in a startup".

Instructor Notes

1. Being a data scientist in an organization is often a chaotic experience, and it's the data scientist's job to try to create order from that chaos. So I wanted to simulate that chaotic experience for my students throughout the semester. But I also wanted them to know that things were going to be slightly chaotic for a pedagogical reason, and not due to my ineptitude!
2. I wanted to draw out different interpretations of the word "chaos" as a means to think about the importance of vocabulary, and the difficulties caused in communication when people either don't know what a word means, or have different ideas of what the word means. Data scientists might be communicating with domain experts who don't really understand what "logistic regression" means, say, but will pretend to know because they don't want to appear stupid, or because they think they ought to know, and therefore don't ask. But then the whole conversation is not really a successful communication if the two people talking don't really understand what they're talking about. Similarly, the data scientists ought to be asking questions to make sure they understand the terminology the domain expert is using (be it an astrophysicist, a social networking expert, or a climatologist). There's nothing wrong with not knowing what a word means, but there is something wrong with not asking! You will likely find that asking clarifying questions about vocabulary gets you even more insight into the underlying data problem.

3. Simulation is a useful technique in data science. It can be useful practice to simulate fake datasets from a model to understand the generative process better, for example, and also to debug code.

Case Study: RealDirect

Doug Perlson, the CEO of RealDirect, has a background in real estate law, startups, and online advertising. His goal with RealDirect is to use all the data he can access about real estate to improve the way people sell and buy houses.

Normally, people sell their homes about once every seven years, and they do so with the help of professional brokers and current data. But there's a problem both with the broker system and the data quality. RealDirect addresses both of them.

First, the brokers. They are typically "free agents" operating on their own—think of them as home sales consultants. This means that they guard their data aggressively, and the really good ones have lots of experience. But in the grand scheme of things, that really means they have only slightly more data than the inexperienced brokers.

RealDirect is addressing this problem by hiring a team of licensed real-estate agents who work together and pool their knowledge. To accomplish this, it built an interface for sellers, giving them useful data-driven tips on how to sell their house. It also uses interaction data to give real-time recommendations on what to do next.

The team of brokers also become data experts, learning to use information-collecting tools to keep tabs on new and relevant data or to access publicly available information. For example, you can now get data on co-op (a certain kind of apartment in NYC) sales, but that's a relatively recent change.

One problem with publicly available data is that it's old news—there's a three-month lag between a sale and when the data about that sale is available. RealDirect is working on real-time feeds on things like when people start searching for a home, what the initial offer is, the time between offer and close, and how people search for a home online.

Ultimately, good information helps both the buyer and the seller. At least if they're honest.

How Does RealDirect Make Money?

First, it offers a subscription to sellers—about $395 a month—to access the selling tools. Second, it allows sellers to use RealDirect's agents at a reduced commission, typically 2% of the sale instead of the usual 2.5% or 3%. This is where the magic of data pooling comes in: it allows RealDirect to take a smaller commission because it's more optimized, and therefore gets more volume.

The site itself is best thought of as a platform for buyers and sellers to manage their sale or purchase process. There are statuses for each person on site: active, offer made, offer rejected, showing, in contract, etc. Based on your status, different actions are suggested by the software.

There are some challenges they have to deal with as well, of course. First off, there's a law in New York that says you can't show all the current housing listings unless those listings reside behind a registration wall, so RealDirect requires registration. On the one hand, this is an obstacle for buyers, but serious buyers are likely willing to do it. Moreover, places that don't require registration, like Zillow, aren't true competitors to RealDirect because they are merely showing listings without providing any additional service. Doug pointed out that you also need to register to use Pinterest, and it has tons of users in spite of this.

RealDirect comprises licensed brokers in various established realtor associations, but even so it has had its share of hate mail from realtors who don't appreciate its approach to cutting commission costs. In this sense, RealDirect is breaking directly into a guild. On the other hand, if a realtor refused to show houses because they are being sold on RealDirect, the potential buyers would see those listings elsewhere and complain. So the traditional brokers have little choice but to deal with RealDirect even if they don't like it. In other words, the listings themselves are sufficiently transparent so that the traditional brokers can't get away with keeping their buyers away from these houses.

Doug talked about key issues that a buyer might care about—nearby parks, subway, and schools, as well as the comparison of prices per square foot of apartments sold in the same building or block. This is the kind of data they want to increasingly cover as part of the service of RealDirect.

Exercise: RealDirect Data Strategy

You have been hired as chief data scientist at *realdirect.com*, and report directly to the CEO. The company (hypothetically) does not yet have its data plan in place. It's looking to you to come up with a data strategy. Here are a couple ways you could begin to approach this problem:

1. Explore its existing website, thinking about how buyers and sellers would navigate through it, and how the website is structured/organized. Try to understand the existing business model, and think about how analysis of RealDirect user-behavior data could be used to inform decision-making and product development. Come up with a list of research questions you think could be answered by data:

   - What data would you advise the engineers log and what would your ideal datasets look like?
   - How would data be used for reporting and monitoring product usage?
   - How would data be built back into the product/website?

2. Because there is no data yet for you to analyze (typical in a start-up when its still building its product), you should get some auxiliary data to help gain intuition about this market. For example, go to *https://github.com/oreillymedia/doing_data_science*. Click on Rolling Sales Update (after the fifth paragraph).

   You can use any or all of the datasets here—start with Manhattan August, 2012–August 2013.

   - First challenge: load in and clean up the data. Next, conduct exploratory data analysis in order to find out where there are outliers or missing values, decide how you will treat them, make sure the dates are formatted correctly, make sure values you think are numerical are being treated as such, etc.
   - Once the data is in good shape, conduct exploratory data analysis to visualize and make comparisons (i) across neighborhoods, and (ii) across time. If you have time, start looking for meaningful patterns in this dataset.

3. Summarize your findings in a brief report aimed at the CEO.

4. Being the "data scientist" often involves speaking to people who aren't also data scientists, so it would be ideal to have a set of communication strategies for getting to the information you need about the data. Can you think of any other people you should talk to?
5. Most of you are not "domain experts" in real estate or online businesses.
    - Does stepping out of your comfort zone and figuring out how you would go about "collecting data" in a different setting give you insight into how you do it in your own field?
    - Sometimes "domain experts" have their own set of vocabulary. Did Doug use vocabulary specific to his domain that you didn't understand ("comps," "open houses," "CPC")? Sometimes if you don't understand vocabulary that an expert is using, it can prevent you from understanding the problem. It's good to get in the habit of asking questions because eventually you will get to something you do understand. This involves persistence and is a habit to cultivate.
6. Doug mentioned the company didn't necessarily have a data strategy. There is no industry standard for creating one. As you work through this assignment, think about whether there is a set of best practices you would recommend with respect to developing a data strategy for an online business, or in your own domain.

Sample R code

Here's some sample R code that takes the Brooklyn housing data in the preceding exercise, and cleans and explores it a bit. (The exercise asks you to do this for Manhattan.)

```
# Author: Benjamin Reddy

require(gdata)
bk <- read.xls("rollingsales_brooklyn.xls",pattern="BOROUGH")
head(bk)
summary(bk)

bk$SALE.PRICE.N <- as.numeric(gsub("[^[:digit:]]","",
                              bk$SALE.PRICE))
count(is.na(bk$SALE.PRICE.N))

names(bk) <- tolower(names(bk))
```

```
## clean/format the data with regular expressions
bk$gross.sqft <- as.numeric(gsub("[^[:digit:]]","",
                            bk$gross.square.feet))
bk$land.sqft <- as.numeric(gsub("[^[:digit:]]","",
                           bk$land.square.feet))

bk$sale.date <- as.Date(bk$sale.date)
bk$year.built <- as.numeric(as.character(bk$year.built))

## do a bit of exploration to make sure there's not anything
## weird going on with sale prices
attach(bk)

hist(sale.price.n)
hist(sale.price.n[sale.price.n>0])
hist(gross.sqft[sale.price.n==0])

detach(bk)

## keep only the actual sales
bk.sale <- bk[bk$sale.price.n!=0,]

plot(bk.sale$gross.sqft,bk.sale$sale.price.n)
plot(log(bk.sale$gross.sqft),log(bk.sale$sale.price.n))

## for now, let's look at 1-, 2-, and 3-family homes
bk.homes <- bk.sale[which(grepl("FAMILY",
            bk.sale$building.class.category)),]
plot(log(bk.homes$gross.sqft),log(bk.homes$sale.price.n))

bk.homes[which(bk.homes$sale.price.n<100000),]
    [order(bk.homes[which(bk.homes$sale.price.n<100000),]
    $sale.price.n),]

## remove outliers that seem like they weren't actual sales
bk.homes$outliers <- (log(bk.homes$sale.price.n) <=5) + 0
bk.homes <- bk.homes[which(bk.homes$outliers==0),]

plot(log(bk.homes$gross.sqft),log(bk.homes$sale.price.n))
```