JAY JACOBS + BOB RUDIS

Data-Driven **Security**

Analysis, Visualization and Dashboards

WILEY

Data-Driven Security: Analysis, Visualization and Dashboards

Published by
John Wiley & Sons, Inc.
10475 Crosspoint Boulevard
Indianapolis, IN 46256
www.wiley.com

Copyright © 2014 by John Wiley & Sons, Inc., Indianapolis, Indiana

Published by John Wiley & Sons, Inc., Indianapolis, Indiana

Published simultaneously in Canada

ISBN: 978-1-118-79372-5

ISBN: 978-1-118-79366-4 (ebk)

ISBN: 9789-1-118-79382-4 (ebk)

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

Library of Congress Control Number: 2013954100

Contents

Introduction

"It's a dangerous business, Frodo, going out your door. You step onto the road, and if you don't keep your feet, there's no knowing where you might be swept off to."

Bilbo Baggins, *The Fellowship of the Ring*

In recent years, cybersecurity has taken center stage in the personal and professional lives of the majority of the global population. Data breaches are a daily occurrence, and intelligent adversaries target consumers, corporations, and governments with practically no fear of being detected or facing consequences for their actions. This is all occurring while the systems, networks, and applications that comprise the backbones of commerce and critical infrastructure are growing ever more complex, interconnected, and unwieldy.

Defenses built solely on the elements of faith-based security—unaided intuition and "best" practices—are no longer sufficient to protect us. The era of the security shaman is rapidly fading, and it's time to adopt the proven tools and techniques being used in other disciplines to take an evolutionary step into *Data-Driven Security*.

Overview of the Book and Technologies

Data-Driven Security: Analysis, Visualization and Dashboards has been designed to take you on a journey into the world of security data science. The start of the journey looks a bit like the word cloud shown in Figure 1, which was created from the text in the chapters of this book. You have a great deal of information available to you, and *may* be able to pick out a signal or two within the somewhat hazy noise on your own. However, it's like looking for a needle in a haystack without a magnet.

project http relatively
representation code make
attributes SIMPLE combination require
really BREACH MEASURE REGRESSION following algorithm FIGURE know
report VERIS validation set interactive little
present TEST features communicate work perform machine
EVENTS CLUSTERING line categories SECTION framework
graphic COLOR HELP DASHBOARD POINTS bit USE NEED TOOLS indicators multiple
definitely level development understand executive output significant
SYSTEMS value type DATA different answer start NODES
CLASSIFICATION important possible focus fields Python correlation
STATE information quantitative variables RESEARCH selection interesting
malware support infections number generate LIST count
OPTIONS display CALCULATE COMPARE FUNCTION available ORGANIZATION natural actually
begin based elements PROVIDE DRIVEN questions want relationship apply simply
QUERY APPLICATION OPERATIONS come security THINGS MAP DISTRIBUTION EXPLORE book
installed people identify skills analysts linear BETTER STANDARD
benefit DESCRIPTIVE SINGLE VISUALIZATION quite internet example chapter good country population
movements observations plot AlienVault
VARIATION basic group way STATISTICS analysis vulnerabilities record model complex graph build
going think network LEARNING objects ATTEMPT
completely consumer details time designed especially
change effective prediction INCIDENT ll SPECIFIC include NORMAL
new structure experience LOOK run CREATE process coefficient chart column large
domain language database order consider limited MEAN reputation management CHALLENGE PATTERNS
problem industry TECHNIQUES ADDRESSES APPROACH hosts
probably individual methods collect environment STRAIGHTFORWARD
world ACTION results represent table program TRAINING
components relational

Figure 1

You'll have much more success identifying what matters (see Figure 2) if you apply the right tools in the most appropriate way possible.

Figure 2

This book focuses on Python and R as the foundational data analysis tools, but also introduces the design and creation of modern static and interactive visualizations with HTML5, CSS, and JavaScript. It also provides background on and security use cases for modern NoSQL databases.

How This Book Is Organized

Rather than have you gorge at an all-you-can-eat buffet, the chapters are more like tapas—each with their own distinct flavor profiles and textures. Like the word *tapas* itself suggests, each chapter covers a different foundational topic within security data science and provides plenty of pointers for further study.

Chapter 1 lays the foundation for the journey and provides examples of how other disciplines have evolved into data-driven practices. It also provides an overview of the skills a security data scientist needs.

Chapters 2, 3, and 4 dive right into the tools, technologies, and basic techniques that should be part of every security data scientists' toolbox. You'll work with AlienVault's IP Reputation database (one of the most thorough sources of malicious nodes publicly available) and take a macro look at the ZeuS and ZeroAccess botnets. We introduce the analytical side of Python in Chapters 2 and 3. Then we thrust you into the world of statistical analysis software with a *major* focus on the R language in the remainder of the book. Unlike traditional introductory texts in R (or statistics in general), we use security data throughout the book to help make the concepts as real and practical as possible for the information security professional.

Chapter 5 introduces some techniques for creating maps and introduces some core statistical concepts, along with a lesson or two about extraterrestrial visitors.

Chapter 6 delves into the biological and cognitive science foundations of visual communication (data visualization) and even shows you how to animate your security data.

This lays a foundation for learning how to analyze and visualize security breaches in **Chapter 7**, where you'll also have an opportunity to work with real incident data.

Chapter 8 covers modern database concepts with new tricks for traditional database deployments and new tools with a range of NoSQL solutions discussed. You'll also get tips on how to answer the question, "Have we seen this IP address on our network?"

Chapter 9 introduces you to the exciting and relatively new world of machine learning. You'll learn about the core concepts and explore a handful of machine-learning techniques and develop a new appreciation for how algorithms can pick up patterns that your intuition might never recognize.

Chapters 10 and 11 give you practical advice and techniques for building effective visualizations that will both communicate and (hopefully) impress your consumers. You'll use everything from Microsoft Excel to state of the art tools and libraries, and be able to translate what you've learned outside of security. Visualization concepts are made even more tangible through "makeovers" of security dashboards that many of you may be familiar with.

Finally, we show you how to apply what you've learned at both a personal and organizational level in **Chapter 12**.

Who Should Read This Book

We wrote this book because we both thoroughly enjoy working with data and wholeheartedly believe that we can make significant progress in improving cybersecurity if we take the time to understand how to ask the right questions, perform accurate and reproducible analyses on data, and communicate the results in the most compelling ways possible.

Readers will get the most out of this book if they come to it with some security domain experience and the ability to do basic coding or scripting. If you are already familiar with Python, you can skip the introduction to it in Chapter 2 and can skim through much of Chapter 3. We level the field a bit by introducing and focusing on R, but you would do well to make your way through all the examples and listings that use R throughout the book, as it is an excellent language for modern data science. If you are new to programming, Chapters 2, 3, and 4 will provide enough of an immersive experience to help you see if it's right for you.

We place emphasis on statistical and machine learning across many chapters and do not recommend skipping any of that content. However, you *can* hold off on Chapter 9 (which discusses machine learning) until the very end, as it will not detract significantly from the flow of the book.

If you know databases well, you need only review the use cases in Chapter 8 to ensure you're thinking about all the ways you can use modern and specialized databases in security use cases.

Unlike many books that discuss dashboards, the only requirements for Chapter 10 are Microsoft Excel or OpenOffice Calc, as we made no assumptions about the types of tools and restrictions you have to work with in your organization. You can also save Chapter 11 for future reading if you have no desire to build interactive visualizations.

In short, though we are writing to Information Technology and Information Security professionals, students, consultants, and anyone looking for more about the how-to of analyzing data and making it understandable for protecting networks will find what they need in this book.

Tools You Will Need

Everything you need to follow along with the exercises is freely available:

- **The R project** (`http://www.r-project.org`)—Most of the examples are written in R, and with the wide range of community developed packages like ggplot2 (`http://ggplot2.org`) almost anything is possible.
- **RStudio** (`http://www.rstudio.com/`)—It will be *much* easier to get to know R and run the examples if you use the RStudio IDE.
- **Python** (`http://www.python.org`)—A few of the examples leverage Python and with add-on packages like pandas (`http://pandas.pydata.org`) makes this a very powerful platform.
- **Sublime Text** (`http://www.sublimetext.com/`)—This, or another robust text editor, will come in very handy especially when working with HTML/CSS/JavaScript examples.
- **D3.js** (`http://d3js.org/`)—Grabbing a copy of D3 and giving the basics a quick read through ahead of Chapter 11 will help you work through the examples in that chapter a bit faster.
- **Git** (`http://git-scm.com/`)—You'll be asked to use git to download data at various points in the book, so installing it now will save you some time later.
- **MongoDB** (`http://www.mongodb.org/`)—MongoDB is used in Chapter 8, so getting it set up early will make those examples less cumbersome.
- **Redis** (`http://redis.io/`)—This, too, is used in some examples in Chapter 8.
- **Tableau Public** (`http://www.tableausoftware.com/`)—If you intend to work with the survey data in Chapter 11, having a copy of Tableau Public will be useful.

Additionally, all of the code, examples, and data used in this book are available through the companion website for this book (`www.wiley.com/go/datadrivensecurity`).

We recommend using Linux or Mac OS, but all of the examples should work fine on modern flavors of Microsoft Windows as well.

What's on the Website

As mentioned earlier, you'll want to check out the companion website `www.wiley.com/go/datadrivensecurity` for the book, which has the full source code for all code listings, the data files used in the examples, and any supporting documents (such as Microsoft Excel files).

The Journey Begins!

You have everything you need to start down the path to *Data-Driven Security*. We hope your journey will be filled with new insights and discoveries and are confident you'll be able to improve your security posture if you successfully apply the principles you're about to learn.

1

The Journey to Data-Driven Security

"It ain't so much the things we don't know that get us into trouble. It's the things we know that just ain't so."

Josh Billings, Humorist

This book isn't really about data analysis and visualization.

Yes, almost every section is focused on those topics, but being able to perform good data analysis and produce informative visualizations is just a means to an end. You never (okay, rarely) analyze data for the sheer joy of analyzing data. You analyze data and create visualizations to gain new perspectives, to find relationships you didn't know existed, or to simply discover new information. In short, you do data analysis and visualizations to learn, and that is what this book is about. You want to learn how your information systems are functioning, or more importantly how they are failing and what you can do to fix them.

The cyber world is just too large, has too many components, and has grown far too complex to simply rely on intuition. Only by augmenting and supporting your natural intuition with the science of data analysis will you be able to maintain and protect an ever-growing and increasingly complex infrastructure. We are not advocating replacing people with algorithms; we are advocating arming people with algorithms so that they can learn more and do a better job. The data contains information, and you can learn better with the information in the data than without it.

This book focuses on using real data—the types of data you have probably come across in your work. But rather than focus on huge discoveries in the data, this book focuses more on the process and less on the result. As a result of that decision, the use cases are intended to be exemplary and introductory rather than knock-your-socks-off cool. The goal here is to teach you new ways of looking at and learning from data. Therefore, the analysis is intended to be new ground in terms of technique, not necessarily in conclusion.

A Brief History of Learning from Data

One of the best ways of appreciating the power of statistical data analysis and visualization is to look back in history to a time when these methods were first put to use. The following cases provide a vivid picture of "before" versus "after," demonstrating the dramatic benefits of the then-new methods.

Nineteenth Century Data Analysis

Prior to the twentieth century, the use of data and statistics was still relatively undeveloped. Although great strides were made in the eighteenth century, much of the scientific research of the day used basic descriptive statistics as evidence for the validity of the hypothesis. The inability to draw clear conclusions from noisy data (and almost all real data is more or less noisy) made much of the scientific debates more about opinions of the data than the data itself. One such fierce debate[1] in the nineteenth century was between two medical professionals in which they debated (both with data) the cause of cholera, a bacterial infection that was often fatal.

The cholera outbreak in London in 1849 was especially brutal, claiming more than 14,000 lives in a single year. The cause of the illness was unknown at that time and two competing theories from two researchers emerged. Dr. William Farr, a well-respected and established epidemiologist, argued that cholera was caused by air pollution created by decomposing and unsanitary matter (officially called the ***miasma*** theory). Dr. John Snow, also a successful epidemiologist who was not as widely known as Farr, put forth the theory that cholera was spread by consuming water that was contaminated by a "special animal poison" (this was prior to the discovery of bacteria and germs). The two debated for years.

Farr published the "Report on the Mortality of Cholera in England 1848–49" in 1852, in which he included a table of data with eight possible explanatory variables collected from the 38 registration districts of London.

[1] And worthy of a bona fide Hollywood plot as well. See `http://snowthemovie.com/`

In the paper, Farr presented some relatively simple (by today's standards) statistics and established a relationship between the average elevation of the district and cholera deaths (lower areas had more deaths). Although there was also a relationship between cholera deaths and the source of drinking water (another one of the eight variables he gathered), he concluded that it was not nearly as significant as the elevation. Farr's theory had data and logic and was accepted by his peers. It was adopted as fact of the day.

Dr. John Snow was passionate and vocal about his disbelief in Farr's theory and relentless in proving his own. It's said he even collected data by going door to door during the cholera outbreak in the Soho district of 1854. It was from that outbreak and his collected data that he made his now famous map in Figure 1-1. The hand-drawn map of the Soho district included little tick marks at the addresses where cholera deaths were reported. Overlaying the location of water pumps where residents got their drinking water showed a rather obvious clustering around the water pump on Broad Street. With his map and his passionate pleas, the city did allow the pump handle to be removed and the epidemic in that region subsided. However, this wasn't enough to convince his critics. The cause of cholera was heavily debated even beyond John Snow's death in 1858.

The cholera debate included data and visualization techniques (long before computers), yet neither had been able to convince the opposition. The debate between Snow and Farr was re-examined in 2003 when statisticians in the UK evaluated the data Farr published in 1852 with modern methods. They found that the data Farr pointed to as proof of an airborne cause actually supported Snow's position. They concluded that if modern statistical methods were available to Farr, the data he collected would have changed his conclusion. The good news of course, is that these statistical methods are available today to you.

Twentieth Century Data Analysis

A few years before Farr and Snow debated cholera, an agricultural research station north of London at Rothamsted began conducting experiments on the effects of fertilizer on crop yield. They spent decades conducting experiments and collecting data on various aspects such as crop yield, soil measurements, and weather variables. Following a modern-day logging approach, they gathered the data and diligently stored it, but they were unable to extract the full value from it. In 1919 they hired a brilliant young statistician named Ronald Aylmer Fisher to pore through more than 70 years of data and help them understand it. Fisher quickly ran into a challenge with the data being confounded, and he found it difficult to isolate the effect of the fertilizer from other effects, such as weather or soil quality. This challenge would lead Fisher toward discoveries that would forever change not just the world of statistics, but almost every scientific field in the twentieth century.

What Fisher discovered (among many revolutionary contributions to statistics) is that if an experiment was designed correctly, the influence of various effects could not just be separated, but also could be measured and their influence calculated. With a properly designed experiment, he was able to isolate the effects of weather, soil quality, and other factors so he could compare the effects of various fertilizer mixtures. And this work was not limited to agriculture; the same techniques Fisher developed at Rothamsted are still used widely today in everything from medical trials to archaeology dig sites. Fisher's work, and the work of his peers, helped revolutionize science in the twentieth century. No longer could scientists simply collect and present their data as evidence of their claim as they had in the eighteenth century. They now had the tools to design robust experiments and the techniques to model how the variables affected their experiment and observations.

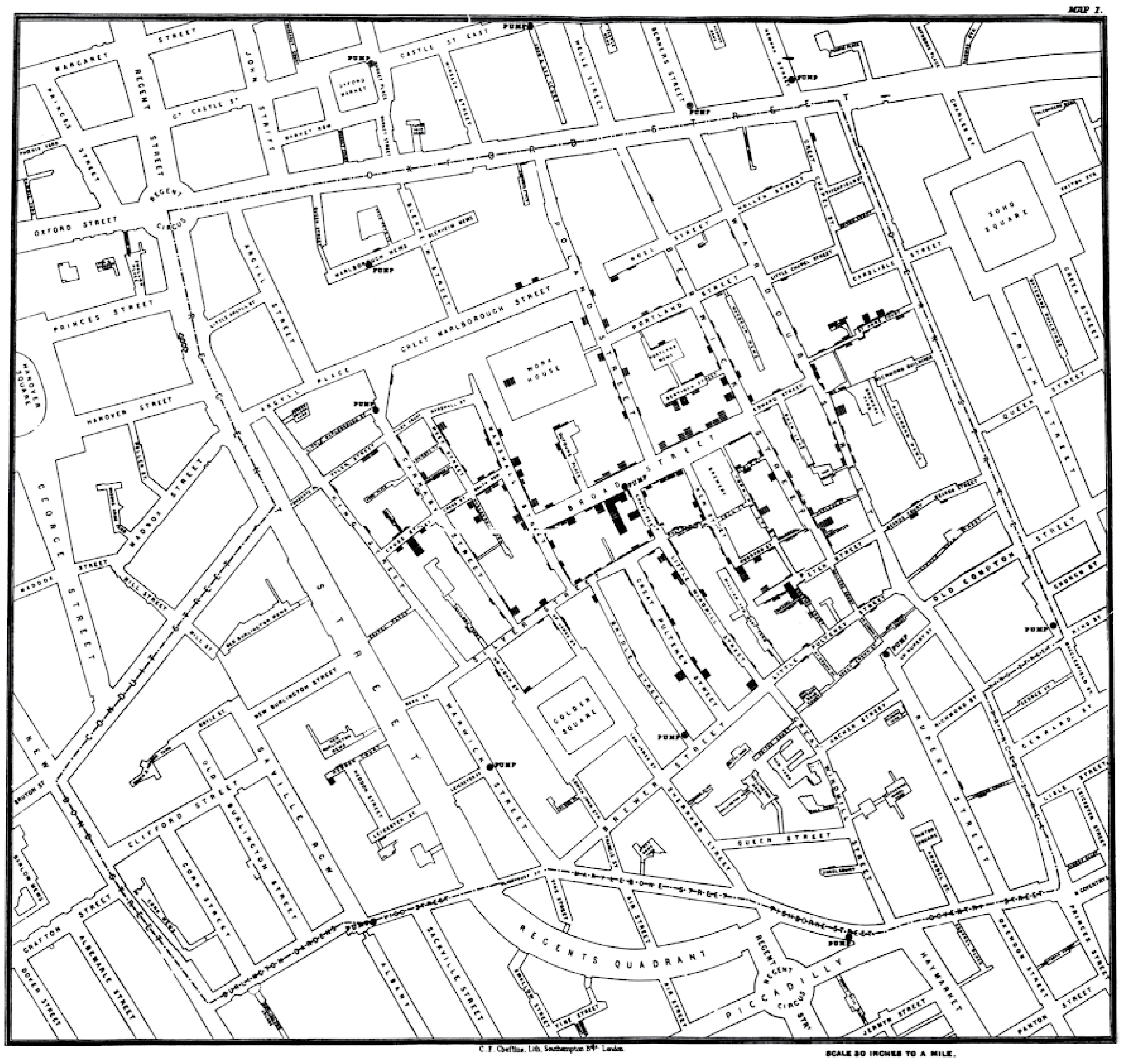

FIGURE 1-1 *Hand-drawn map of the areas affected by cholera*

At this point, the world of science included statistical models. Much of the statistical and science education focused on developing and testing these models and the assumptions behind them. Nearly every statistical problem started with the question—"What's the model?"—and ended with the model populated to allow description and even prediction using the model. This represented a huge leap forward and enabled research never before possible. If it weren't for computers, the world would probably still consider these techniques to be modern. But computers are ubiquitous and they have enabled a whole new approach to data analysis that was both impossible and unfathomable prior to their development.

Twenty-First Century Data Analysis

It's difficult to pull out any single person or event that captures where data analysis is today like Farr and Fisher captured the previous stages of data analysis. The first glimpse at what was on the horizon came from John Tukey, who wrote in 1962 that data analysis should be thought of as different from statistics (although analysis leveraged statistics). He stated that data analysis must draw from science more than mathematics (can you see the term "data science" in there?). Tukey was not only an accomplished statistician, having contributed numerous procedures and techniques to the field, but he was also an early proponent of visualization techniques for the purpose of describing and exploring the data. You will come back to some of Tukey's work later in this chapter.

Let's jump ahead to a paper written in 2001 by Leo Breiman, a statistician who focused on machine learning algorithms (which are discussed in Chapter 9). In the paper he describes a new culture of data analysis that does not focus on defining a data model ***of nature*** but instead derives an algorithmic model ***from nature***. This new culture has evolved within computer science and engineering largely outside (or perhaps alongside) traditional statistics. New approaches are born from the practical problems created by the information age, which created large quantities of complex and noisy data. The revolutionary idea that Breiman outlined in this paper is that models should be judged on their predictive accuracy instead of validating the model with traditional statistical tests (which are not without value by the way).

At face value you may think of testing "predictive accuracy" by gathering data today and determining how it predicts the world of tomorrow, but that's not what the idea is about. The idea is about splitting the data of today into two data sets, using the first data set to generate (or "train") an algorithm and then validating (or "test") its predictive accuracy on the second data set. To increase the power of this approach, you can iterate through this process multiple times, splitting the data into various training and test sets, generating and validating as you go. This approach is not well suited to small data sets, but works remarkably well with modern data sets.

There are several main differences between data analysis in the modern information age and the agricultural fields of Rothamsted. First, there is a large difference in the available sample size. "Classic" statistical techniques were largely limited by what the computers of the day could handle ("computers" were the people hired to "compute" all day long). With generally smaller samples, generating a training and test was impractical. However, modern environments are recording hundreds of variables generated across thousands of systems. Large sample sizes are the norm, not the exception.

Second, for many environments and industries, a properly designed experiment is unlikely if not completely impossible. You cannot divide your networks into control and test groups, nor would you want to test the efficacy of a web application firewall by only protecting a portion of a critical application. One effect of these environmental limits is a much higher noise-to-signal ratio in the data. The techniques of machine learning (and the related field of data mining) have evolved with the challenges of modern data in mind.

Finally, knowledge of statistics is just one skill of many that contributes to successful data analysis in the twenty-first century. With that in mind, the next section spends some time looking at the various skills and attributes that support a good data analysis.

Gathering Data Analysis Skills

We know there is a natural allure to data science and everyone wants to achieve that sexy mystique surrounding security data analysis. Although we have focused on this concept of data analysis so far, it takes more than just analytic skills to create the mystique that everyone is seeking. You need to combine statistics and data analysis with visualization techniques, and then leverage the computing power and mix with a healthy dose of domain (information security) knowledge. All of this begins not with products or tools but with your own skills and abilities.

We don't have the data. An alternate form of this objection is saying that we don't have actuarial-quality data (which is more prevalent when you start talking about risk analysis). Data detractors argue that anything less than perfect data is worthless and prevents you from creating well-designed experiments. This statement is untrue and quite harmful. If you were to wait around for perfect data, you would always be waiting and many learning opportunities would be missed. More importantly and to the heart of this objection, you don't ***need*** perfect data. You just need methods to learn from the messy data you do have. As Douglas Hubbard wrote in 2010 in his book *How to Measure Anything,* "The fact is that we often have more data than we think, we need less data than we think, and getting more data through observation is simpler than we think." So, generally speaking, data for security analysis absolutely exists; often times it is just waiting to be collected. You can, with a few alterations, collect and accurately analyze even sketchy data. Modern data analysis methods have evolved to work with the noisy, incomplete, and imperfect data you have.

But we will fall off the edge of the world. There is one last point to consider and it's not so much an objection to data analysis, but an obstacle in data analysis. When you are seen as a domain expert, you are expected to provide answers with confidence. The conflict arises when confidence is confused with certainty. Data analysis requires just enough self-awareness and humility to create space for doubt in the things you think you know. Even though you may confidently state that passwords should be so many characters long with a certain amount of complexity, the reality is you just don't know where the balance is between usability and security. Confidence needs to be balanced with humility and the ability to update your beliefs based on new evidence. This obstacle in data analysis is not just limited to the primary analyst. Other domain experts involved in the analysis will have to come face to face with their own humility. Not everyone will want to hear that his or her world isn't flat.

Programming Skills

As much as we'd like to portray data science as a glamorous pursuit of truth and knowledge, as we've said, it can get a little messy. Okay, that's an understatement. Working with data is a great deal more uncertain and unkempt than people think and, unfortunately, the mess usually appears early on when you're attempting to collect and prepare the data. This is something that many classes in statistics never prepare their students for. Professors hand out rather nice and neat data sets ready to be imported into the analysis tool *du jour.* Once you leave the comfort of the classroom, you quickly realize that the world is a disorganized and chaotic place and data (and its subsequent analyses) are a reflection of that fact.

This is a cold, hard lesson in data science: Data comes to you in a wide range of formats, states, and quality. It may be embedded in unstructured or semi-structured log files. It may need to be scraped from a website. Or, in extreme cases, data may come in an overly complex and thoroughly frustrating format known as XML. Somehow, you must find a way to collect, coax, combine, and massage what you're given into a format that supports further analysis. Although this could be done with a lot of patience, a text editor, and judicious use of summer interns, the ability to whip together a script to do the work will provide more functionality, flexibility, and efficiency in the long run. Learning even basic programming skills opens up a whole range of possibilities when you're working with data. It frees you to accept multiple forms of data and manipulate it into whatever formats work best with the analysis software you have. Although there is certainly a large collection of handy data conversion tools available, they cannot anticipate or handle everything you will come across. To be truly effective while working with data, you need to adapt to the data in your world, not vice versa.

AES-256-Bit Keys Are Twice as Good as AES-128, Right?

One natural assumption about AES-256-bit keys is that because they are twice as long as AES-128-bit keys, they are twice as secure. We've been around information security people when they force a project to use 256-bit keys because they are "twice as good." Well, let's look into the math. First, you are talking about bits here, and although 256 bits is twice as many bits as 128, 256-bit keys actually have 2^{128} ***times*** more keys. Break out your slide rules and work through an exercise to try to answer a simple question: If you had access to the world's fastest super-computer, how many 128-bit keys could you crack?

The world's fastest super computer (at the time of this writing) is the ***Tianhe-2*** in China, which does around 34 petaflops (34×10^{15} floating point operations) per second. If you assume it takes one operation to generate a key and one operation to test it (this is an absurd and conservative assumption), you can test an amazing 17×10^{15} keys per second. But a 128-bit key has 3.4×10^{38} possibilities, which means after a full year of cracking 128-bit keys, you will have exhausted 1.6×10^{-13} percent of the key space. Even if you run the super-computer for 1,000 years, you will only have searched 0.0000000000016 percent of all the possible keys (and spent a fortune on electricity).

To put this simply, ***the probability of brute-force cracking a 128-bit key is already so infinitesimally small that you could easily round off that probability to zero***. But let's be professional here and say, "Moving from a 128-bit key to a 256 is moving the probability from really-super-duper-infinitesimally-small to really-super-duper-infinitesimally-small x 2^{128}."

Any modern language will support basic data manipulation tasks, but scripting languages such as Python and R appear to be used slightly more often in data analysis than their compiled counterparts (Java and C). However, the programming language is somewhat irrelevant. The end results (and a happy analyst) are more important than picking any "best" language. Whatever gets the job done with the least amount of effort is the best language to use. We generally flip between Python (pandas) and R for cleaning and converting data (or perhaps some Perl if we're feeling nostalgic) and then R or pandas for the analysis and visualization. Learning web-centric languages like HTML, CSS, and JavaScript will help create interactive visualizations for the web, as you'll see in Chapter 11, but web languages are not typically involved in the preparation and analysis of data.

There is a tool worth mentioning in this section—the "gateway tool" between a text editor and programming—known as the ***spreadsheet*** (such as Microsoft Excel or OpenOffice Calc). Spreadsheets allow non-programmers to do some amazing things and get some quick and accessible results. Although spreadsheets have their own sets of challenges and drawbacks, they also have some benefits. If the data is not too large or complex and the task is not deciding the future of the world economy (see the following sidebar), Excel may be the best tool for the job. We strongly suggest seeing Excel as a temporary solution. It does well at quick one-shot tasks. But if you have a repeating analytic task or model that is used repeatedly, it's best to move to some type of structured programming language.

As a cleaning tool, spreadsheets seem like a very good solution at first (especially for those who have developed some skill with them). But spreadsheets are event-driven, meaning they work through clicking, typing, and dragging. If you want to apply a conversion to a row of data, you have to click to select the row and apply a conversion. This works for small data sets or quick tasks, but trust us, you will (more often than you think) have to go back to the source data and re-clean it. Another day of log files needs to be processed, or you realize you should have pulled another relationship from the source data, or (gasp) you identify an error in the cleaning process. Something, somewhere, and probably more than once, will cause you to go back to the source and repeat the data cleaning and conversion. Leveraging a spreadsheet means a lot more clicking. Writing a script, on the other hand, enables an easy, flexible, and consistent execution of the cleaning process each time it runs.

The Limits of Spreadsheets

On January 16th, 2013, J.P. Morgan issued a report to shareholders titled "Report of JPMorgan Chase & Co. Management Task Force Regarding 2012 CIO Losses" (full citation in Appendix B) in which they investigate the loss of $6 billion in trades. They perform a detailed examination of the breakdown and describe the spreadsheet as a contributory factor. "During the review process, additional operational issues became apparent. For example, the model operated through a series of Excel spreadsheets, which had to be completed manually, by a process of copying and pasting data from one spreadsheet to another." They uncovered a huge challenge with spreadsheets, which is the consistency and integrity of the computations made in the data. "Data were uploaded manually without sufficient quality control. Spreadsheet-based calculations were conducted with insufficient controls and frequent formula and code changes were made." They continue on and label the Excel-based model as "error prone" and "not easily scalable." As with any complex system, catastrophe requires multiple failures.[2] We cannot point to their use of an "error-prone" spreadsheet as the primary cause, but certainly it appears to have contributed in the loss of $6 billion.

[2] See Richard Cook's "How Complex Systems Fail" for a brief and wonderful discussion of this topic: `http://www.ctlab.org/documents/How%20Complex%20Systems%20Fail.pdf`

After the data is ready for analysis, you can continue to benefit from understanding how to program. Many of the languages mentioned here have robust data analysis features built into (or onto) them. For example, statisticians developed the R language specifically for the purpose of performing data analysis. Python—with the addition of packages like NumPy, SciPy, and pandas—offers a rich and comparable data analysis environment. But, preparing and analyzing the data is not enough. You also need to communicate your results, and one of the most effective methods for that is data visualization (covered in several chapters of this book). Again, Excel can produce graphics. With judicial modification of the default settings, you can get good visualization with Excel. However, in our opinion, flexibility and detail in data visualization are best achieved through programming. Both Python and R have some feature-rich packages for generating and exporting data visualization. In many cases, however, you can combine all these steps and functions in the same script. You can write one script to grab the source data, manipulate/clean it, run the analysis on it, and then visualize the results.

Data Management

If there is one skill you can hold off on learning, it's data management, but you can put it off only for a while. Within information security (as well as most other disciplines), your data can quickly multiply. If you don't learn to manage it, the strain of ever-expanding data will take its toll on efficiency and effectiveness. As mentioned, you can leverage spreadsheets for the simple analyses. You will quickly outgrow that stage and should be resolved to expanding your repertoire to programming languages and simple formats like comma-separated value (CSV) files. At this point, you may see some benefits by moving your data into a database, but it still may not be necessary.

As the data repository grows, you reach a tipping point, either through the complexity of the data or the volume of data. Moving to a more robust data management solution becomes inevitable. There is a misconception that the large relational databases of yesteryear are reserved for the biggest projects, and that is not a helpful mindset. Many of the database systems discussed in Chapter 8 can be installed on a desktop and make the analysis more efficient and scalable. Once your data management skills become more natural, such skill can benefit even the smallest projects. We've installed a local database and imported the data even for some smaller one-time projects.

When discussing data management skills, we naturally focus on databases. You want to have enough knowledge to install a relational or NoSQL database to dump the data in and leverage it for analysis. However, data management is more than databases. Data management is also about managing the quality and integrity of the data. You want to be sure the data you are working with isn't inadvertently modified or corrupted. It doesn't hurt to have some checks that keep an eye on data quality and integrity, especially over long-term data analysis efforts (metrics). It's like the concept of unit tests in software development where the smallest piece of testable code in an application is isolated from the larger body of code and checked to determine whether it behaves exactly as expected. You may want to automate some data integrity checking after any new import or conversion, especially when the data analysis has sufficient efficacy to be performed regularly and used as a metric or control.

Finally, we work in information security, and we'd be negligent if we didn't talk about the security of the data for a bit here. Take a step back for some context first. There seems to be a pattern repeating: Some passionate need drives a handful of geniuses to work their tails off to produce an elegant solution, but the security of their system is not their primary concern; meeting the functional need is. As an example, when the UNIX platform was first developed it was intended to be a shared (but closed) platform for multiple users who use the platform for programs they would write. As a result, most of the authentication and permissions were constructed to protect the system from unintentional errors in their programs, and not from malicious users.[3] The point here is that "young" technology typically places an emphasis on functionality over security.

With the fast-paced and passionate push of the current data revolution, we are definitely seeing more emphasis on functionality and less on security. New data management platforms such as Hadoop and NoSQL environments were designed to solve a data problem and were not designed (initially) with many of the security policies or compliance requirements of most enterprise networks (though they are quickly learning). The result is a distributed computing platform with some difficult security challenges. The authentication and security features are far better than the early days of UNIX; they typically do not compare to the security and features of the more established relational databases. We won't focus too much on this point, but whatever data management platform is chosen, don't assume the security is built in.

[3] For an example of the focus on functionality and preventing error over stopping misuse, early authentication systems would store the user passwords in a clear text file. See Morris and Thompson, 1979 (full reference in Appendix B) for a discussion.

Statistics

Perhaps we are a little biased here, but picking up some statistics skills will improve almost every aspect of your life. Not only will it change the way to see and learn from the world around you, but it will also make you more interesting and probably even a bit more attractive to those around you. Seriously, though, statistics (we are discussing it as a single skill here) is a very broad topic and quite a deep well to drink from. We use the term to describe the varied collection of techniques and methods that have evolved (and continue to evolve) to attempt to learn from data. These skills include the classic statistical approaches as well as newer techniques like data mining and machine learning. Luckily, you can learn from the successes and mistakes of the generations of rather brilliant people who have worked with data very similar to ours, even if their calculations were performed with pencil and paper versus silicon circuits. Regardless of your personal belief in the utility of statistics and data analysis, when it comes to information security, there is a vast amount of evidence showing its significant influence and benefit to almost every other field of science.

Aside from the obvious "learning from data" approach, there are a few perhaps more subtle reasons to focus on improving your statistics skills:

- **Even though data never lies, it is far too easy to be tricked by it**—As heuristic beings, we are capable of pulling out patterns and meaning from the world around us. The ability to see subtle connections and patterns is usually helpful, and people use that skill on a daily basis. However, that skill can also mislead you, and you may think you see patterns and connections when none exist. A good understanding of statistics can raise awareness of this, and its tactics can help minimize incorrect conclusions.
- **Even though we just said that data never lies, the way it's generated and collected can create deceptive conclusions**—Consider that asking for the opinions of those around us may mistakenly confirm our own opinions, because we naturally surround ourselves with like-minded people. Data itself may not be deceptive, but it's quite easy to think the data means something it does not, as in the story of the 1936 election polling (see the following sidebar).

Statistics is not just a collection of tools; it is a collection of toolboxes each with their own set of tools. You can begin with descriptive statistics, which attempt to simplify the data into numbers that describe aspects of the data. For example, you can calculate the center of the data by calculating the mean, mode, or median; you can describe how spread out the data is with the standard deviation; you can explain the symmetry of the data with skew; and you can describe the width of peak with the kurtosis. However, any time you simplify the data, you lose some level of detail and this is where visualization can serve you well. With visualizations, you create a single representation, or message, that can contain and communicate every data point, without simplification. Think of this type of visualization as being a "descriptive visualization" since it is doing nothing more than simply describing the data to its viewers.

Aside from the challenge of oversimplifying, descriptive statistics is also limited to describing only the data that you collect. It is not correct to simply scan a few systems, calculate the mean number of vulnerabilities, and announce that the statistic describes all the systems in the environment. Inferential statistics helps you go beyond just describing the observations and enables you to make statements about a larger population given a smaller representative sample from that population. The key word there is "representative." Statistics teaches you about the "design of experiments" (thanks to Fisher and his peers) and this will help you gather data so that you reduce the probability of being misled by it. You want to have confidence that the samples you collect are representative of the whole. That lesson has been learned many times in the past by a good number of people.

These graphs are generated as a way to understand certain relationships and attributes of the model. They communicate from the data to the analyst and are used to visually inspect for anomalies, strength of relationships, or other aspects of the data for the purpose of understanding it better. Very little effort is spent on making these attractive or presentable since they are part of the analysis, not the result.

The other type of visualization exists to communicate from the analyst to others and serves to explain the story (or the lack of a story) the analyst uncovered in the data. These are typically intended to be attractive and carry a clear message, as it is a communication tool for non-analysts. Figure 1-3 (which you'll learn to generate in Chapter 5) is derived from the same data as Figure 1-2 but is intended for a completely different audience. Therefore, it is cleaner and you can pull a message for each of the 48 continental states from this one picture.

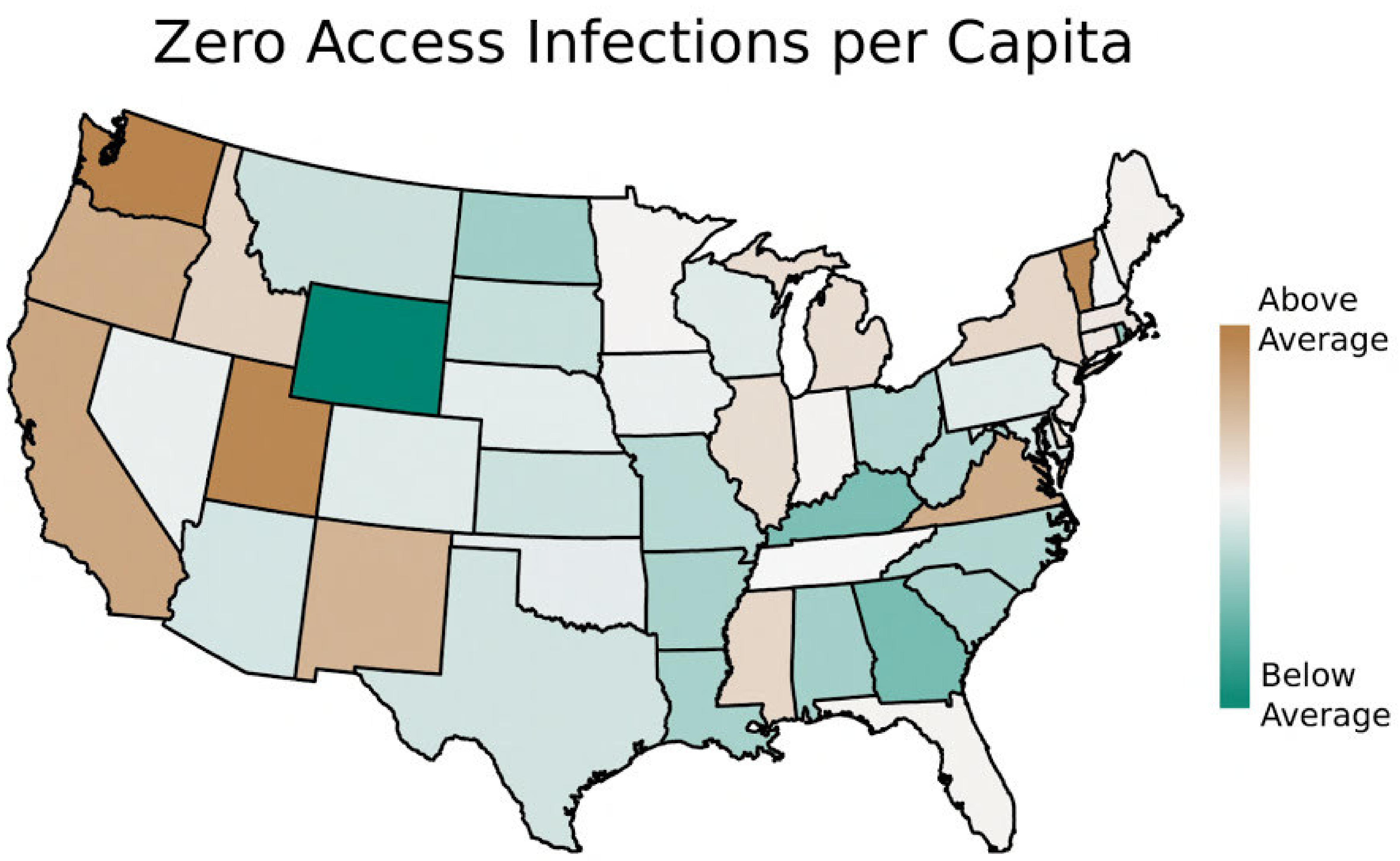

FIGURE 1-3 *Visualization for communicating density of ZeroAccess bot infections*

Combining the Skills

You need some combination of skills covered in this chapter in order to make the analysis run smoother and improve what you can learn from the data. Although we may have portrayed these skills as belonging to a single person, that is not required. As the data grow and the demands for analysis become more embedded into the culture, spreading the load among multiple individuals will help lighten the load. Moreover, if you are just beginning to build your security data science team, you may be setting yourself up for an impossible task if you try to find even one individual with all these skills. Take the time to talk through each of these points with candidates to ensure there is at least some element of each of the skills discussed here.

Centering on a Question

While we consider data analysis to be quite fun, it is never performed for its own sake. Data analysis is always performed within a larger context and understanding that context is the key to successful data analysis. Losing sight of that context is like running a race without paying attention to where the finish line is. You want to have a good concept of what you're trying to learn from the data. Therefore, every good data analysis project begins by setting a goal and creating one or more ***research questions***. Perhaps you have come across a visualization or research and thought, "Yeah, but so what?" That reaction is probably caused by the lack of a well-prepared research question in the analysis. Remember, the purpose of data analysis is to learn from the environment; learning can be done with or without data (with varying degrees of success). Creating and following a good research question is a component of ***good learning***, not just of good data analysis. Without a well-formed question guiding the analysis, you may waste time and energy seeking convenient answers in the data, or worse, ***you may end up answering a question that nobody was asking in the first place***.

For example, Figure 1-4 shows the amount and categories of spam blocked at an organization during a given month. Thanks to the logs generated by an email filtering system, it is entirely possible to collect and show this information. However, the questions this data answers (and whatever subsequent actions it may drive) are of little interest to the typical organization. It's hard to imagine someone looking at this graphic and thinking, "Let's understand why travel spam was up in December." Outcomes like those shown in Figure 1-4 are the result of poor question selection or skipping a question altogether—data analysis for the sake of analyzing data, which does not help to inform anyone about the environment in any meaningful way.

A good research question around spam might be, "How much time do employees spend on spam that is not blocked by the spam filter?" Just counting how much spam is blocked has little value since it will have no contextual meaning (nobody can internalize the effective difference between 1,000 and 5,000 spam emails). What you want to know is the impact spam has on employee productivity. Although "productivity" may be a challenge to measure directly, you can flip that around and just assume it is impossible to be productive when employees are reading and deleting spam. Therefore, what you really want to measure is the time employees spend dealing with unfiltered spam.

Now that you've framed the question like this, it's clear that you can't look to the spam filter logs to answer this spam-related question. You really don't care that thousands of emails were blocked at the perimeter or even what proportion of spam is blocked. With a research question in hand, you now know to collect a measurement of employee time. Perhaps you can look for logs from the email clients of events when users select the "mark as spam" option. Or perhaps, it's important enough to warrant running a short survey in which you select a sample of users and ask them to record the amount of spam and time spent going through it for some limited period of time. Either way, the context and purpose of the analysis is being set by the research question, not by the availability of data.

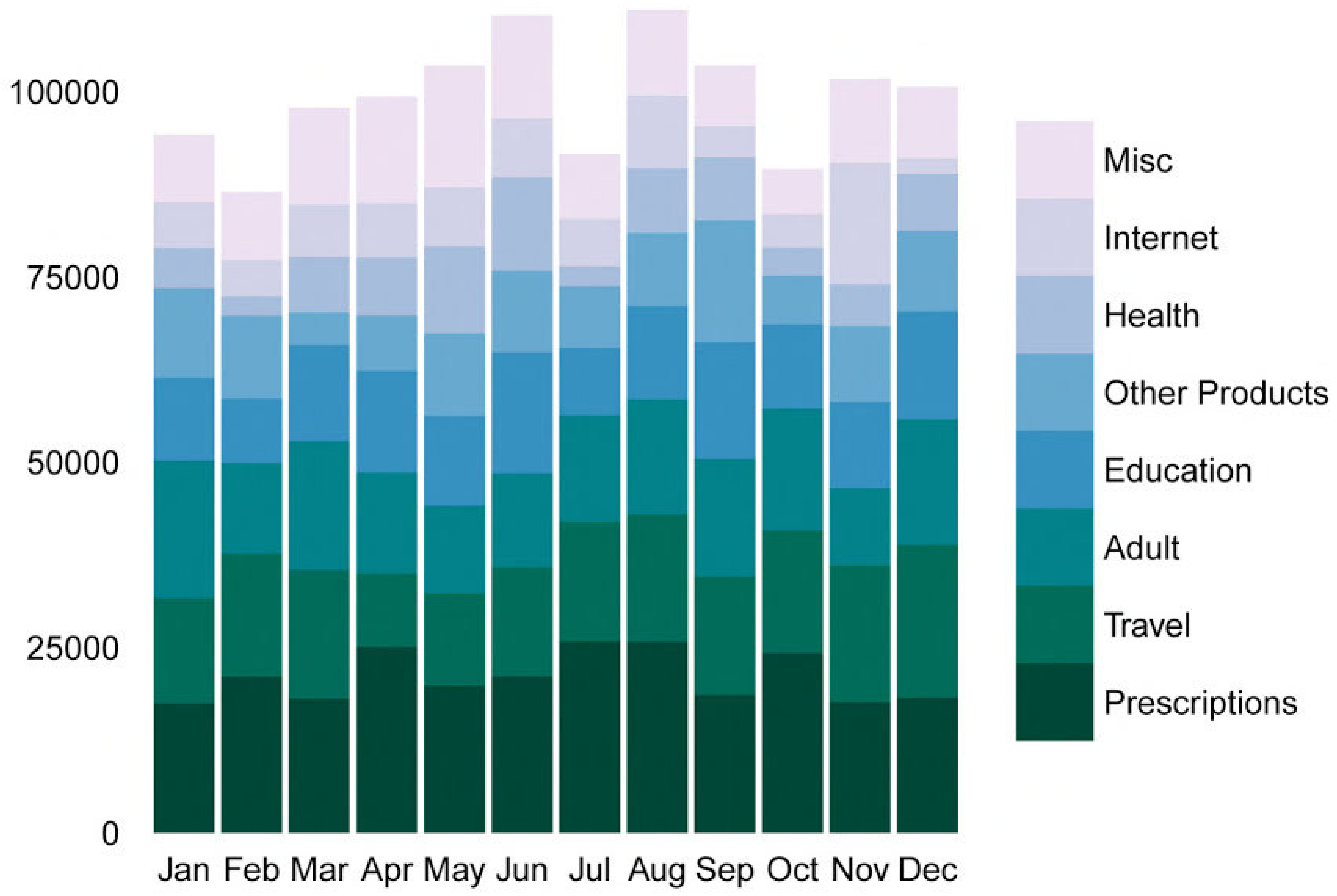

Figure 1-4 *Amount of spam by category—the result of a poor research question*

Creating a Good Research Question

Creating a good research question is relatively straightforward but requires a bit of practice, critical thinking, and discipline. Most research questions will serve as pivot points for a decision or action (or inaction). Knowing the context of the result may also help determine what to collect. Going back to the spam example, maybe you learn there is some tolerance for wasted time. If so, maybe you don't need to how much time is wasted, but just whether the time spent dealing with spam is simply above or below that tolerance. Planning the analysis with that information could change how data is sought or simplify data storage and analysis.

You usually begin with some topic already in mind. Perhaps you are measuring the possible benefit from a technical change or you are trying to protect a specific asset or data type, or simply trying to increase your visibility into a network segment. Even if you just have a general sense of direction, you can begin by coming up with a series of questions or things you'd like to know about it. Once you have a good list of questions, you can whittle those down to one or just a few related questions. Now the fun really begins—you have to make those questions objective.

Consider this simple example. The Human Resources department submits a proposal to post a searchable lunch menu from the company's cafeteria to the Internet. Although this may raise all sorts of questions around controls, processes, and procedures, suppose the core security-oriented decision of the proposal is limited to either allowing authentication with the corporate username and password, or investing in a more expensive two-factor authentication mechanism. You may brainstorm a question like "How much risk does single factor authentication represent?" Or perhaps, "How effective is two-factor authentication?" These types of questions are really nice and squishy for the initial phase of forming a research question, but not well suited to serious analysis. You would struggle to collect evidence of "risk" or "effectiveness" in these questions. So, you must transform them to be more specific and measurable as an approach to inform the decisions or actions in context. Perhaps you start by asking how many services require single-factor versus dual-factor authentication. You might also like to know how many of those services have been attacked, and with what success, and so on. Perhaps you have access to a honey pot and can research and create a profile of Internet-based brute force attempts. Perhaps you can look at the corporate instance of Microsoft Outlook Web Access and create a profile of authentication-based attacks on that asset. These are all good questions that are very answerable with data analysis. They can produce outcomes that can help support a decision.

Exploratory Data Analysis

Now that we've explained how a good data analysis should begin, we want to talk about how things will generally occur in the real world. It'd be great to start each day with a hot, caffeinated beverage, a clear research question, and a bucket of clean data, but in reality you'll usually have to settle for just the hot, caffeinated beverage. Often times, you do start off with data and a vague question like, "Is there anything useful in this data?" This brings us back to John Tukey (remember him from earlier in this chapter?). He pioneered a process he called ***exploratory data analysis***, or EDA. It's the process of walking around barefoot in the data, perhaps even rolling around a bit in it. You do this to learn about the variables in the data, their significance, and their relationships to other variables. Tukey developed a whole range of techniques to increase your visibility into and understanding of the data, including the elegantly simple ***stem and leaf plot***, the ***five-number summary***, and the helpful ***box plot*** diagram. Each of these techniques is explained or used later in this book.

Once you get comfortable with the data, you'll naturally start to ask some question of it. However, and this is important, you always want to circle back and form a proper research question. As Tukey said in his 1977 book, "Exploratory data analysis can never be the whole story." He refers to EDA as the foundation stone and the first step in data analysis. He also said that, "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." With that in mind, most of the use cases in this book use exploratory analysis. We will take an iterative approach, and you'll learn as you walk around in the data. In the end though, you need to remember that data analysis is performed to find an answer to a question that's worthy of asking.

Summary

The cyber world is just too large, has too many components, and has grown far too complex to simply rely on intuition. Generations of people before us have paved the way; and with a mixture of domain expertise, programing experience and statistics combined with data management and visualization skills, we can improve on our ability to learn from this complex environment through the data it produces.

In the next chapter we will walk you through setting up your data analysis environment, and then proceed into Chapter 3, where you will be guided through a gentle introduction to data analysis techniques.

Recommended Reading

The following are some recommended readings that can further your understanding on some of the topics we touch on in this chapter. For full information on these recommendations and for the sources we cite in the chapter, please see Appendix B.

"Conditions for Intuitive Expertise: A Failure to Disagree" by Daniel Kahneman and Gary Klein—This dense article covers a lot of ground but gets at the heart of when and why you should look for help in complex environments and when your expertise is enough. The references in this paper also provide a good jumping point to answer questions about how people learn.

"How Complex Systems Fail" by Richard Cook—If you are wondering whether or not you are dealing with complexity, this short and brilliant paper looks at qualities of complex systems and how they fail.

***Naked Statistics: Stripping the Dread from Data* by Charles Wheelan**—This is a great introductory book to statistical concepts and approaches, written in an easy-to-consume style and written so that the math is not required (but it is included).

Before you jump right into the various use cases in the book, it's important to ensure you at least have a basic familiarity with the two most prominent languages featured in nearly all of the scenarios: Python (`www.python.org/`) and R (`www.r-project.org/`). Although there are an abundance of tools available for data analysis, we feel these two provide virtually all the features necessary to help you go from data to discovery with the least amount impedance.

A sub-theme throughout the book, and the distilled process at the heart of security data science, is ***idea***, ***exploration***, ***trial*** (and ***error***) and ***iteration***. It is ineffective at best to attempt to shoehorn this process into the ***edit***/***compile***/***run*** workflow found in most traditional languages and development environments. The acts of performing data analyses and creating informative visualizations are highly interactive and iterative endeavors. Despite all of their positive features, even standalone Python and R do not truly enable rich, dynamic interaction with code and data. However, when they are coupled with IPython (`http://ipython.org/`) and RStudio (`www.rstudio.com/`), respectively, they are transformed into powerful exploration tools, enabling rapid development and testing of everything from gnarly data munging to generating sophisticated visualizations.

This chapter provides pointers to installation resources for each tool, introduces core features of each language and development environment, and explains the structure of the examples you will find in the remaining chapters of the book. Each chapter will have the following "setup" code (Listing 2-0) at the beginning to ensure you have the proper environment in place to run the code examples. There are example scripts at the end of this chapter that will help you create structured directories if you are typing as you go.

Listing 2-0

```
# This is for the R code in the chapter
# set working directory to chapter location
# (change for where you set up files in ch 2)
setwd("~/book/ch02")
# This is for the Python code in the chapter
# loads the necessary Python library for chdir
import os
# set working directory to chapter location
os.chdir(os.path.expanduser("~") + "/book/ch02")
```

Why Python? Why R? And Why Both?

A discussion of which programming language is better than another for a certain set of tasks often turns (quickly) into a religious war of words that rarely wins converts and never becomes fully resolved. As a security data scientist, you will find that you do not have the luxury of language bias. There will be times when one language shines in one area while a different one shines in another, and you need the skills of a diplomat to bring them both together to solve real problems.

We've honed in on both R/RStudio and Python/IPython/pandas in this book, as they are the two leading data analysis languages/environments with broad similarities but also with unique elements that make them work well for some tasks and not others. As you read about the rationale behind each choice and as you become proficient in one or both environments, do not lull yourself into a sense of complacency.

For readers with an existing programming background, getting up to speed with Python should be pretty straightforward and you can expect to be fairly proficient within 3–6 months, especially if you convert some of your existing scripts over to it as a learning exercise. Your code may not be "pythonic" (that is, utilizing the features, capabilities, and the syntax of the language in the most effective way), but you will be able to "get useful stuff done." For those who are new to statistical languages, becoming proficient in R may pose more of a challenge. Statisticians created R, and that lineage becomes fairly obvious as you delve into the language. If you can commit to suffering through R syntax and package nuances, plus commit to transitioning some of your existing Excel workflows into R, you too should be able to hang with the cool kids on the `#rstats` Twitter stream in 3–6 months.

Note

A hallmark of a good data scientist is adaptability and you should be continually scouring the digital landscape for emerging tools that will help you solve problems. We introduce you to some of these upstarts in Appendix A.

Why Python?

Guido van Rossum created the Python programming language in December of 1989 to solve a problem. He and his colleagues needed a common way to orchestrate system administration tasks that could take advantage of specific features in the operating systems they were using at that time. Although there were existing interpreted, administrator-friendly tools and languages available, none were designed (from Guido van Rossum's point of view) with either the flexibility or extensibility features baked into the design principles of Python.

Python's flexibility and extensibility (and the fact that it was free as in both "speech" and "beer") were especially appealing to the scientific, academic, and industrial communities starting in the early 2000s. Innovators in these fields quickly adapted this general-purpose programming language to their own disciplines to solve problems easier than—ostensibly—the domain-specific languages available at that time.

You have to search long and hard to find a file-type Python cannot read, a database Python cannot access, and an algorithm Python cannot execute. As you familiarize yourself with the language, Python's ability to acquire, clean, and transform source data will quickly amaze you, but those tasks are just the early steps in your analysis and visualization process. It wasn't until 2008 that the pandas (`http://pandas.pydata.org/`) module was created by AQR Capital Management to provide "Pythonic" counterparts to the analytical foundations of languages like R, SAS, or MATLAB, which is where the "real fun" begins.

Although Python's interpreter provides an interactive execution shell, aficionados recognized the need to extend this basic functionality and developed an even more dynamic and robust interactive environment—IPython—to fill the need. When coupled with the pandas module, budding data analysts now have a mature and data-centric toolset available to drive their quest for knowledge.

Why R?

Unlike Python, R's history is inexorably tied to its domain specific predecessors and cousins, as it is 100 percent focused and built for statistical data analysis and visualization. Although it too can access and manipulate various file types and databases (and was also designed for flexibility and extensibility), R's lisp- and S-like syntax plus extreme focus on foundational analytics-oriented data types has kept it, mostly, in the hands of the "data crunchers."

Base R makes it remarkably simple to run extensive statistical analyses on your data and then generate informative and appealing visualizations with just a few lines of code. More modern R libraries such as `plyr` and `ggplot2` extend and enhance these base capabilities and are the foundations of many of mind- and eye-catching examples of cutting-edge data analysis and visualization you have no doubt come across on the Internet.

Like Python, R also provides an interactive execution shell that has enough basic functionality for general needs. Yet, the desire for even more interactivity sparked the development of RStudio, which is a combination of integrated development environment (IDE), data exploration tool, and iterative experimentation environment that exponentially enhances R's default capabilities.

Why Both?

If all you have is a hammer, everything starts looking like a nail. There are times when the flexibility of a general-purpose programming language comes in very handy, which is when you use Python. There are other times when three lines of R code will do something that may take 30 or more lines of Python code (even with pandas) to accomplish. Since your ultimate goal is to provide insightful and accurate analyses as quickly and as visually appealing as possible, knowing which tool to use for which job is a critical insight you must develop to be as effective and efficient as possible.

We would be a bit dishonest, though, if we did not concede that there are some things that Python can do (easily or at all) that R cannot, and vice-versa. We touch upon some of these in the use cases throughout the book, but many of the—ah—"learning opportunities" will only come from performing your own analyses, getting frustrated (which is the polite way of saying "stuck"), and finding resolution by jumping to another tool to "get stuff done." This situation comes up frequently enough that there is even an `rJython` package for R that lets you call Python code from R scripts, and `rpy` and `rpy2` modules for Python that let you call R code from Python scripts.

By having both tools in your toolbox, you should be able to tackle most, if not all, of the tasks that come your way. If you do find yourself in a situation where you need functionality you don't have, both R and Python have vibrant communities that are eager to provide assistance and even help in the development of new functions or modules to fit emerging needs.

Jumpstarting Your Python Analytics with Canopy

It ***is*** possible to set up an effective and efficient installation of Python, IPython, and pandas from the links we've provided, especially if you are already familiar or proficient with Python; however, we don't recommend it. For those new to Python, the base installation leaves you with the core interpreter and extensive set of built-in, standard libraries. You can think of it as a having an inexpensive blank canvas and introductory set of paints and brushes. You'll need better materials to create a work of art, and that's where the enhanced statistics, computational and graphing libraries come in. Even the most stalwart Python aficionado can find it challenging to manage dependencies and updates for the numerous necessary components. This can waste hours of your time. This is especially true if you have to manage analytics processes across multiple operating systems and environments.