CHAPTER

# 1

# What Is Data Science?



Sometimes, understanding what something ***is*** includes having a clear picture of what it ***is not***. Understanding data science is no exception. Thus, this chapter begins by investigating what data science is not, because the term has been much abused and a lot of hype surrounds big data and data science. You will first consider the difference between true data science and fake data science. Next, you will learn how new data science training has evolved from traditional university degree programs. Then you will review several examples of how modern data science can be used in real-world scenarios.

Finally, you will review the history of data science and its evolution from computer science, business optimization, and statistics into modern data science and its trends. At the end of the chapter, you will find a Q&A section from recent discussions I've had that illustrate the conflicts between data scientists, data architects, and business analysts.

This chapter asks more questions than it answers, but you will find the answers discussed in more detail in subsequent chapters. The purpose of this approach is for you to become familiar with how data scientists think, what is important in the big data industry today, what is becoming obsolete, and what people interested in a data science career don't need to learn. For instance, you need to know statistics, computer science, and machine learning, but not everything from these domains. You don't need to know the details about complexity of sorting algorithms (just the general results), and you don't need to know how

to compute a generalized inverse matrix, nor even know what a generalized inverse matrix is (a core topic of statistical theory), unless you specialize in the numerical aspects of data science.

**TECHNICAL NOTE**

**This chapter can be read by anyone with minimal mathematical or technical knowledge. More advanced information is presented in "Technical Notes" like this one, which may be skipped by non-mathematicians.**

**CROSS-REFERENCE** **You will find definitions of most terms used in this book in Chapter 8.**

## Real Versus Fake Data Science

Books, certificates, and graduate degrees in data science are spreading like mushrooms after the rain. Unfortunately, many are just a mirage: people taking advantage of the new paradigm to quickly repackage old material (such as statistics and R programming) with the new label "data science."

Expanding on the R programming example of fake data science, note that R is an open source statistical programming language and environment that is at least 20 years old, and is the successor of the commercial product S+. R was and still is limited to in-memory data processing and has been very popular in the statistical community, sometimes appreciated for the great visualizations that it produces. Modern environments have extended R capabilities (the in-memory limitations) by creating libraries or integrating R in a distributed architecture, such as RHadoop (R + Hadoop). Of course other languages exist, such as SAS, but they haven't gained as much popularity as R. In the case of SAS, this is because of its high price and the fact that it was more popular in government organizations and brick-and-mortar companies than in the fields that experienced rapid growth over the last 10 years, such as digital data (search engine, social, mobile data, collaborative filtering). Finally, R is not unlike the C, Perl, or Python programming languages in terms of syntax (they all share the same syntax roots), and thus it is easy for a wide range of programmers to learn. It also comes with many libraries and a nice user interface. SAS, on the other hand, is more difficult to learn.

To add to the confusion, executives and decision makers building a new team of data scientists sometimes don't know exactly what they are looking for, and they end up hiring pure tech geeks, computer scientists, or people lacking proper big data experience. The problem is compounded by Human Resources

(HR) staff who do not know any better and thus produce job ads that repeat the same keywords: Java, Python, MapReduce, R, Hadoop, and NoSQL. But is data science really a mix of these skills?

Sure, MapReduce is just a generic framework to handle big data by reducing data into subsets and processing them separately on different machines, then putting all the pieces back together. So it's the distributed architecture aspect of processing big data, and these farms of servers and machines are called the *cloud*.

Hadoop is an implementation of MapReduce, just like C++ is an implementation (still used in finance) of object oriented programming. NoSQL means "Not Only SQL" and is used to describe database or data management systems that support new, more efficient ways to access data (for instance, MapReduce), sometimes as a layer hidden below SQL (the standard database querying language).

**CROSS-REFERENCE** **See Chapter 2 for more information on what MapReduce can't do.**

There are other frameworks besides MapReduce — for instance, graph databases and environments that rely on the concepts of nodes and edges to manage and access data, typically spatial data. These concepts are not necessarily new. Distributed architecture has been used in the context of search technology since before Google existed. I wrote Perl scripts that perform *hash joins* (a type of NoSQL join, where a join is the operation of joining or merging two tables in a database) more than 15 years ago. Today some database vendors offer hash joins as a fast alternative to SQL joins. Hash joins are discussed later in this book. They use hash tables and rely on *name-value pairs*. The conclusion is that MapReduce, NoSQL, Hadoop, and Python (a scripting programming language great at handling text and unstructured data) are sometimes presented as Perl's successors and have their roots in systems and techniques that started to be developed decades ago and have matured over the last 10 years. But data science is more than that.

Indeed, you can be a real data scientist and have none of these skills. NoSQL and MapReduce are not new concepts — many people embraced them long before these keywords were created. But to be a data scientist, you also need the following:

- Business acumen
- Real big data expertise (for example, you can easily process a 50 million-row data set in a couple of hours)
- Ability to sense the data
- A distrust of models
- Knowledge of the curse of big data
- Ability to communicate and understand which problems management is trying to solve

- Ability to correctly assess lift — or ROI — on the salary paid to you
- Ability to quickly identify a simple, robust, scalable solution to a problem
- Ability to convince and drive management in the right direction, sometimes against its will, for the benefit of the company, its users, and shareholders
- A real passion for analytics
- Real applied experience with success stories
- Data architecture knowledge
- Data gathering and cleaning skills
- Computational complexity basics — how to develop robust, efficient, scalable, and portable architectures
- Good knowledge of algorithms

A data scientist is also a generalist in business analysis, statistics, and computer science, with expertise in fields such as robustness, design of experiments, algorithm complexity, dashboards, and data visualization, to name a few. Some data scientists are also data strategists — they can develop a data collection strategy and leverage data to develop actionable insights that make business impact. This requires creativity to develop analytics solutions based on business constraints and limitations.

The basic mathematics needed to understand data science are as follows:

- Algebra, including, if possible, basic matrix theory.
- A first course in calculus. Theory can be limited to understanding computational complexity and the *O* notation. Special functions include the logarithm, exponential, and power functions. Differential equations, integrals, and complex numbers are not necessary.
- A first course in statistics and probability, including a familiarity with the concept of random variables, probability, mean, variance, percentiles, experimental design, cross-validation, goodness of fit, and robust statistics (not the technical details, but a general understanding as presented in this book).

From a technical point a view, important skills and knowledge include R, Python (or Perl), Excel, SQL, graphics (visualization), FTP, basic UNIX commands (sort, grep, head, tail, the pipe and redirect operators, cat, cron jobs, and so on), as well as a basic understanding of how databases are designed and accessed. Also important is understanding how distributed systems work and where bottlenecks are found (data transfers between hard disk and memory, or over the Internet). Finally, a basic knowledge of web crawlers helps to access unstructured data found on the Internet.

## Two Examples of Fake Data Science

Here are two examples of fake data science that demonstrate why data scientists need a standard and best practices for their work. The two examples discussed here are not bad products — they indeed have a lot of intrinsic value — but they are not data science. The problem is two-fold:

- First, statisticians have not been involved in the big data revolution. Some have written books about applied data science, but it's just a repackaging of old statistics courses.
- Second, methodologies that work for big data sets — as big data was defined back in 2005 when 20 million rows would qualify as big data — fail on post-2010 big data that is in terabytes.

As a result, people think that data science is statistics with a new name; they confuse data science and fake data science, and big data 2005 with big data 2013. Modern data is also very different and has been described by three *V*s: velocity (real time, fast flowing), variety (structured, unstructured such as tweets), and volume. I would add veracity and value as well. For details, read the discussion on when data is flowing faster than it can be processed in Chapter 2.

**CROSS-REFERENCE** **See Chapter 4 for more detail on statisticians versus data scientists.**

### *Example 1: Introduction to Data Science e-Book*

Looking at a 2012 data science training manual from a well-known university, most of the book is about old statistical theory. Throughout the book, R is used to illustrate the various concepts. But logistic regression in the context of processing a mere 10,000 rows of data is not big data science; it is fake data science. The entire book is about *small* data, with the exception of the last few chapters, where you learn a bit of SQL (embedded in R code) and how to use an R package to extract tweets from Twitter, and create what the author calls a word cloud (it has nothing to do with cloud computing).

Even the Twitter project is about small data, and there's no distributed architecture (for example, MapReduce) in it. Indeed, the book never talks about data architecture. Its level is elementary. Each chapter starts with a short introduction in simple English (suitable for high school students) about big data/data science, but these little data science excursions are out of context and independent from the projects and technical presentations.

Perhaps the author added these short paragraphs so that he could rename his "Statistics with R" e-book as "Introduction to Data Science." But it's free and it's a nice, well-written book to get high-school students interested in statistics and programming. It's just that it has nothing to do with data science.

### *Example 2: Data Science Certificate*

Consider a data science certificate offered by a respected public university in the United States. The advisory board is mostly senior technical guys, most having academic positions. The data scientist is presented as "a new type of data analyst." I disagree. Data analysts include number crunchers and others who, on average, command lower salaries when you check job ads, mostly because these are less senior positions. Data scientist is not a junior-level position.

This university program has a strong data architecture and computer science flair, and the computer science content is of great quality. That's an important part of data science, but it covers only one-third of data science. It also has a bit of old statistics and some nice lessons on robustness and other statistical topics, but nothing about several topics that are useful for data scientists (for example, Six Sigma, approximate solutions, the 80/20 rule, cross-validation, design of experiments, modern pattern recognition, lift metrics, third-party data, Monte Carlo simulations, or the life cycle of data science projects. The program does requires knowledge of Java and Python for admission. It is also expensive — several thousand dollars.

So what comprises the remaining two-thirds of data science? Domain expertise (in one or two areas) counts for one-third. The final third is a blend of applied statistics, business acumen, and the ability to communicate with decision makers or to make decisions, as well as vision and leadership. You don't need to know everything about six sigma, statistics, or operations research, but it's helpful to be familiar with a number of useful concepts from these fields, and be able to quickly find good ad hoc information on topics that are new to you when a new problem arises. Maybe one day you will work on time-series data or econometric models (it happened unexpectedly to me at Microsoft). It's okay to know a little about time series today, but as a data scientist, you should be able to identify the right tools and models and catch up very fast when exposed to new types of data. It is necessary for you to know that there is something called *time series*, and when faced with a new problem, correctly determine whether applying a time-series model is a good choice or not. But you don't need to be an *expert* in time series, Six Sigma, Monte Carlo, computational complexity, or logistic regression. Even when suddenly exposed to (say) time series, you don't need to learn everything, but you must be able to find out what is important by doing quick online research (a critical skill all data scientists should have). In this case (time series), if the need arises, learn about correlograms, trends, change point, normalization and periodicity. Some of these topics are described in Chapter 4 in the section Three Classes Of Metrics: Centrality, Volatility, Bumpiness.

## The Face of the New University

Allow me to share two stories with you that help to illustrate one of the big problems facing aspiring data scientists today. I recently read the story of an

adjunct professor paid $2,000 to teach a class, but based on the fee for the course and the number of students, the university was earning about $50,000 from that class. So where does the $48,000 profit go?

My wife applied for a one-year graduate program that costs $22,000. She then received a letter from the university saying that she was awarded a $35,000 loan to pay for the program. But if she needed a loan to pay for the program, she would not have pursued it in the first place.

The reason I share these two stories is to point out that the typically high fees for U.S. graduate and undergraduate programs are generally financed by loans, which are causing a student debt crisis is the United States. The assumption is that traditional universities charge such high fees to cover equally high expenses that include salaries, facilities, operations, and an ever-growing list of government regulations with which they must comply. Because of this, traditional universities are facing more and more competition from alternative programs that are more modern, shorter, sometimes offered online on demand, and *cost much less* (if anything).

**NOTE** **Not all countries have educational costs as high as those encountered in the United States. It could make sense for an American student to attend a data science program abroad — for instance, in China or Europe — to minimize costs while still having access to high-quality material and gaining international experience.**

Since we are criticizing the way data science is taught in some traditional curricula, and the cost of traditional university educations in the United States, let's think a bit about the future of data science higher education.

Proper training is fundamental, because that's how you become a good, qualified data scientist. Many new data science programs offered online (such as those at Coursera.com) or by corporations (rather than universities) share similar features, such as being delivered online, or on demand. Here is a summary regarding the face of the new data science "university."

The new data science programs are characterized by the following:

- Take much less time to earn, six months rather than years
- Deliver classes and material online, on demand
- Focus on applied modern technology
- Eliminate obsolete content (differential equations or eigenvalues)
- Include rules of thumb, tricks of the trade, craftsmanship, real implementations, and practical advice integrated into training material
- Cost little or nothing, so no need to take on large loans
- Are sometimes sponsored or organized by corporations and/or forward-thinking universities (content should be vendor-neutral)

- No longer include knowledge silos (for instance, operations research versus statistics versus business analytics)
- Require working on actual, real-world projects (collaboration encouraged) rather than passing exams
- Include highly compact, well-summarized training material, pointing to selected free online resources as necessary
- Replace PhD programs with apprenticeships
- Provide substantial help in finding a good, well paid, relevant job (fee and successful completion of program required; no fee if program sponsored by a corporation: it has already hired or will hire you)
- Are open to everyone, regardless of prior education, language, age, immigration status, wealth, or country of residence
- Are even more rigorous than existing traditional programs
- Have reduced cheating or plagiarism concerns because the emphasis is not on regurgitating book content
- Have course material that is updated frequently with new findings and approaches
- Have course material that is structured by focusing on a vertical industry (for instance, financial services, new media/social media/advertising), since specific industry knowledge is important to identifying and understanding real-world problems, and being able to jump-start a new job very quickly when hired (with no learning curve)

Similarly, the new data science "professor" has the following characteristics:

- Is not tenured, yet not an adjunct either
- In many cases is not employed by a traditional university
- Is a cross-discipline expert who constantly adapts to change, and indeed brings meaningful change to the program and industry
- Is well connected with industry leaders
- Is highly respected and well known
- Has experience in the corporate world, or experience gained independently (consultant, modern digital publisher, and so on)
- Publishes research results and other material in online blogs, which is a much faster way to make scientific progress than via traditional trade journals
- Does not spend a majority of time writing grant proposals, but rather focuses on applying and teaching science
- Faces little if any bureaucracy

- Works from home in some cases, eliminating the dual-career location problem faced by PhD married couples
- Has a lot of freedom in research activities, although might favor lucrative projects that can earn revenue
- Develops open, publicly shared knowledge rather than patents, and widely disseminates this knowledge
- In some cases, has direct access to market
- Earns more money than traditional tenured professors
- Might not have a PhD

**CROSS-REFERENCE** **Chapter 3 contains information on specific data science degree and training programs.**

## The Data Scientist

The data scientist has a unique role in industry, government, and other organizations. That role is different from others such as statistician, business analyst, or data engineer. The following sections discuss the differences.

### Data Scientist Versus Data Engineer

One of the main differences between a data scientist and a data engineer has to do with ETL versus DAD:

- **ETL** (Extract/Load/Transform) is for data engineers, or sometimes data architects or database administrators (DBA).
- **DAD** (Discover/Access/Distill) is for data scientists.

Data engineers tend to focus on software engineering, database design, production code, and making sure data is flowing smoothly between source (where it is collected) and destination (where it is extracted and processed, with statistical summaries and output produced by data science algorithms, and eventually moved back to the source or elsewhere). Data scientists, while they need to understand this data flow and how it is optimized (especially when working with Hadoop) don't actually optimize the data flow itself, but rather the data processing step: extracting value from data. But they work with engineers and business people to define the metrics, design data collecting schemes, and make sure data science processes integrate efficiently with the enterprise data systems (storage, data flow). This is especially true for data scientists working in small companies, and is one reason why data scientists should be able to write code that is re-usable by engineers.

Sometimes data engineers do DAD, and sometimes data scientists do ETL, but it's not common, and when they do it's usually internal. For example, the data engineer may do a bit of statistical analysis to optimize some database processes, or the data scientist may do a bit of database management to manage a small, local, private database of summarized information.

DAD is comprised of the following:

- **Discover**: Identify good data sources and metrics. Sometimes request the data to be created (work with data engineers and business analysts).
- **Access**: Access the data, sometimes via an API, a web crawler, an Internet download, or a database access, and sometimes in-memory within a database.
- **Distill**: Extract from the data the information that leads to decisions, increased ROI, and actions (such as determining optimum bid prices in an automated bidding system). It involves the following:
  - Exploring the data by creating a data dictionary and exploratory analysis
  - Cleaning the data by removing impurities.
  - Refining the data through data summarization, sometimes multiple layers of summarization, or hierarchical summarization)
  - Analyzing the data through statistical analyses (sometimes including stuff like experimental design that can take place even before the Access stage), both automated and manual. Might or might not require statistical modeling
  - Presenting results or integrating results in some automated process

Data science is at the intersection of computer science, business engineering, statistics, data mining, machine learning, operations research, Six Sigma, automation, and domain expertise. It brings together a number of techniques, processes, and methodologies from these different fields, along with business vision and action. Data science is about bridging the different components that contribute to business optimization, and eliminating the silos that slow down business efficiency. It has its own unique core, too, including (for instance) the following topics:

- Advanced visualizations
- Analytics as a Service (AaaS) and API's
- Clustering and taxonomy creation for large data sets
- Correlation and R-squared for big data
- Eleven features any database, SQL, or NoSQL should have
- Fast feature selection
- Hadoop/Map-Reduce
- Internet topology

- Keyword correlations in big data
- Linear regression on an usual domain, hyperplane, sphere, or simplex
- Model-free confidence intervals
- Predictive power of a feature
- Statistical modeling without models
- The curse of big data
- What MapReduce can't do

Keep in mind that some employers are looking for Java or database developers with strong statistical knowledge. These professionals are very rare, so instead the employer sometimes tries to hire a data scientist, hoping he is strong in developing production code. You should ask upfront (during the phone interview, if possible) if the position to be filled is for a Java developer with statistics knowledge, or a statistician with strong Java skills. However, sometimes the hiring manager is unsure what he really wants, and you might be able to convince him to hire you without such expertise if you convey to him the added value your expertise does bring. It is easier for an employer to get a Java software engineer to learn statistics than the other way around.

## Data Scientist Versus Statistician

Many statisticians think that data science is about analyzing data, but it is more than that. Data science also involves implementing algorithms that process data automatically, and to provide automated predictions and actions, such as the following:

- Analyzing NASA pictures to find new planets or asteroids
- Automated bidding systems
- Automated piloting (planes and cars)
- Book and friend recommendations on Amazon.com or Facebook
- Client-customized pricing system (in real time) for all hotel rooms
- Computational chemistry to simulate new molecules for cancer treatment
- Early detection of an epidemic
- Estimating (in real time) the value of all houses in the United States (Zillow.com)
- High-frequency trading
- Matching a Google Ad with a user and a web page to maximize chances of conversion
- Returning highly relevant results to any Google search

- Scoring all credit card transactions (fraud detection)
- Tax fraud detection and detection of terrorism
- Weather forecasts

All of these involve both statistical science and terabytes of data. Most people doing these types of projects do not call themselves statisticians. They call themselves data scientists.

Statisticians have been gathering data and performing linear regressions for several centuries. DAD performed by *statisticians* 300 years ago, 20 years ago, today, or in 2015 for that matter, has little to do with DAD performed by *data scientists* today. The key message here is that eventually, as more statisticians pick up on these new skills and more data scientists pick up on statistical science (sampling, experimental design, confidence intervals — not just the ones described in Chapter 5), the frontier between data scientist and statistician will blur. Indeed, I can see a new category of data scientist emerging: data scientists with strong statistical knowledge.

What also makes data scientists different from computer scientists is that they have a much stronger statistics background, especially in computational statistics, but sometimes also in experimental design, sampling, and Monte Carlo simulations.

## Data Scientist Versus Business Analyst

Business analysts focus on database design (database modeling at a high level, including defining metrics, dashboard design, retrieving and producing executive reports, and designing alarm systems), ROI assessment on various business projects and expenditures, and budget issues. Some work on marketing or finance planning and optimization, and risk management. Many work on high-level project management, reporting directly to the company's executives.

Some of these tasks are performed by data scientists as well, particularly in smaller companies: metric creation and definition, high-level database design (which data should be collected and how), or computational marketing, even *growth hacking* (a word recently coined to describe the art of growing Internet traffic exponentially fast, which can involve engineering and analytic skills).

There is also room for data scientists to help the business analyst, for instance by helping automate the production of reports, and make data extraction much faster. You can teach a business analyst FTP and fundamental UNIX commands: ls -l, rm -i, head, tail, cat, cp, mv, sort, grep, uniq -c, and the pipe and redirect operators (|, >). Then you write and install a piece of code on the database server (the server accessed by the business analyst traditionally via a browser or tools such as Toad or Brio) to retrieve data. Then, all the business analyst has to do is:

1. Create an SQL query (even with visual tools) and save it as an SQL text file.

2. Upload it to the server and run the program (for instance a Python script, which reads the SQL file and executes it, retrieves the data, and stores the results in a CSV file).
3. Transfer the output (CSV file) to his machine for further analysis.

Such collaboration is win-win for the business analyst *and* the data scientist. In practice, it has helped business analysts extract data 100 times bigger than what they are used to, and 10 times faster.

In summary, data scientists are not business analysts, but they can greatly help them, including automating the business analyst's tasks. Also, a data scientist might find it easier get a job if she can bring the extra value and experience described here, especially in a company where there is a budget for one position only, and the employer is unsure whether hiring a business analyst (carrying overall analytic and data tasks) or a data scientist (who is business savvy and can perform some of the tasks traditionally assigned to business analysts). In general, business analysts are hired first, and if data and algorithms become too complex, a data scientist is brought in. If you create your own startup, you need to wear both hats: data scientist and business analyst.

## Data Science Applications in 13 Real-World Scenarios

Now let's look at 13 examples of real-world scenarios where the modern data scientist can help. These examples will help you learn how to focus on a problem and its formulation, and how to carefully assess all of the potential issues — in short, how a data scientist would look at a problem and think strategically before starting to think about a solution. You will also see why some widely available techniques, such as standard regression, might not be the answer in all scenarios.

**NOTE** **This chapter focuses on problems and how to assess them. Chapters 4 and 5 discuss solutions to such problems.**

The *data scientist's* way of thinking is somewhat different from that of engineers, operations research professionals, and computer scientists. Although operations research has a strong analytic component, this field focuses on specific aspects of business optimization, such as inventory management and quality control. Operations research domains include defense, economics, engineering, and the military. It uses Markov models, Monte Carlo simulations, queuing theory, and stochastic process, and (for historical reasons) tools such as Matlab and Informatica.

**CROSS-REFERENCE** **See Chapter 4 for a comparison of data scientists with business analysts, statisticians, and data engineers.**

There are two basic types of data science problems:

1. **Internal data science problems**, such as bad data, reckless analytics, or using inappropriate techniques. Internal problems are not business problems; they are internal to the data science community. Therefore, the fix consists in training data scientists to do better work and follow best practices.
2. **Applied business problems** are real-world problems for which solutions are sought, such as fraud detection or identifying if a factor is a cause or a consequence. These may involve internal or external (third-party) data.

## Scenario 1: DUI Arrests Decrease After End of State Monopoly on Liquor Sales

An article was recently published in the *MyNorthWest* newspaper about a new law that went into effect a year ago in the state of Washington that allows grocery stores to sell hard liquor. The question here is how to evaluate and interpret the reported decline in DUI arrests after the law went into effect.

As a data scientist, you would first need to develop a list of possible explanations for the decline (through discussions with the client or boss). Then you would design a plan to rule out some of them, or attach the correct weight to each of them, or simply conclude that the question is not answerable unless more data or more information is made available.

Following are 15 potential explanations for, and questions regarding, the reported paradox regarding the reported DUI arrest rates. You might even come up with additional reasons.

- There is a glitch in the data collection process (the data is wrong).
- The article was written by someone with a conflict of interest, promoting a specific point of view, or who is politically motivated. Or perhaps it is just a bold lie.
- There were fewer arrests because there were fewer policemen.
- The rates of other crimes also decreased during that timeframe as part of a general downward trend in crime rates. Without the new law, would the decline have been even more spectacular?
- There is a lack of statistical significance.
- Stricter penalties deter drunk drivers.
- There is more drinking by older people and, as they die, DUI arrests decline.
- The population of drinkers is decreasing even though the population in general is increasing, because the highest immigration rates are among Chinese and Indian populations, who drink much less than other population groups.

- Is the decrease in DUI arrests for Washington residents, or for non-residents as well?
- It should have no effect because, before the law, people could still buy alcohol (except hard liquor) in grocery stores in Washington.
- Prices (maybe because of increased taxes) have increased, creating a dent in alcohol consumption (even though alcohol and tobacco are known for their resistance to such price elasticity).
- People can now drive shorter distances to get their hard liquor, so arrests among hard liquor drinkers have decreased.
- Is the decline widespread among all drinkers, or only among hard liquor drinkers?
- People are driving less in general, both drinkers and non-drinkers, perhaps because gas prices have risen.
- A far better metric to assess the impact of the new law is the total consumption of alcohol (especially hard liquor) by Washington residents.

The data scientist must select the right methodology to assess the impact of the new law and figure out how to get the data needed to perform the assessment. In this case, the real cause is that hard liquor drinkers can now drive much shorter distances to get their hard liquor. For the state of Washington the question is, did the law reduce costs related to alcohol consumption (by increasing tax revenue from alcohol sales, laying off state-store employees, or creating modest or no increase in alcohol-related crime, and so on).

## Scenario 2: Data Science and Intuition

Intuition and gut feeling are still what drive many decisions in executive circles. Yet, as demonstrated in this example, data science and statistical analysis are shown to be superior to intuition and keep you from forming wrong conclusions based on a gut feeling.

Twin data points are observations that are almost identical, and they tend to be the norm rather than the exception in many types of data sets. In any 2- or 3-dimensional data set with 300+ rows, if the data is quantitative and evenly distributed in a bounded space, you should expect to see a large proportion (>15 percent) of data points that have a close neighbor.

This applies to all data sets, but the discovery was first made by looking at a picture of stars in a galaxy. There are so many binary stars that you might intuitively assume that there is some mechanism that forces stars to cluster in pairs. However, if you look at pure probabilities, you would see that it is highly probable for 15 percent of all stars to belong to a binary star system without any external mechanism driving it.

For example, consider a galaxy containing 500 stars. Pure probability calculations on how many of these stars would be within binary systems reveal the following:

- The probability of having at least 60 (12 percent) of the stars in a binary system is 85 percent.
- The probability of having at least 80 (16 percent) of the stars in a binary system is 18 percent.
- The probability of having at least 100 (20 percent) of the stars in a binary system is 0 percent (almost).

In reality, however, more than 80 percent of all stars are located within binary systems. This number is not supported by the probability statistics; thus, there is clearly some mechanism at work that forces stars to cluster in pairs.

This framework provides a good opportunity to test your analytic intuition. Look at a chart with twin observations, and visually assess whether the twin observations are natural (random) or not (too numerous or too few of them). It would also be a great exercise to write a piece of code (Python, Perl, or R) that performs these simulations (including the more complicated 3-D case) to double-check the theoretical results and compare R, Perl, and Python in terms of speed.

**CROSS-REFERENCE** **Chapter 6, "Data Science Application Case Studies," contains additional tests you can use to assess your analytic intuition.**

**TECHNICAL NOTE**

**This note explains how the probabilities were computed. You may skip it if you are not a mathematician.**

**Say a night sky image featuring stars is 10 x 10 cm, and has about n=500 visible stars (data points), and a binary star is defined as a star having a neighboring star 1 millimeter away (or less) in the picture. If stars were distributed perfectly randomly, the expected number of stars in a binary star system would be 73 (on average) out of 500 stars. This number is far higher than most people would have thought. You can denote this proportion as *p*, thus p=14.5 percent, and n*p=73, the expected number of stars in a binary system, among these 500 stars.**

**You can compute the probability in question using the theory of stochastic processes — Poisson process in this case. The intensity *L* of the process is the number of points per square millimeter — that is L = 500/(100 mm x 100 mm) = 0.05 per square millimeter, if 500 points were distributed in a 100 x 100 mm area (a magnified picture of the night sky).**

**The probability *p* that a star has at least one neighbor within 1 mm is 1 - Proba(zero neighbor) = 1 - exp(-L*Pi*r^2), where r = 1 mm and Pi = 3.14. Here Pi*r^2 is the area of a circle of radius r = 1 mm. The exponential term comes from the fact that for a Poisson process, the number of points in a given set (circle, rectangle, and so on), has a Poisson distribution of mean L*Area. Thus p=0.145.**

So being a binary star is a Bernoulli (1/0) variable of parameter p=0.145. *V* can denote the number of stars that are in a binary star system: *V* is the sum of *n* Bernoulli variables of parameter *p*, and thus has a Binomial distribution of parameters *n*, *p*. The standardized variable Z = (V - np)/SQRT{np(1-p)} is well approximated by a normal(0,1) distribution. This fact can be used to compute the preceding probabilities.

**Alternative computation**: The same results could have been obtained using Monte Carlo simulations, rather than using a theoretical model, to compute these probabilities. This would have involved the generation of a million simulated images (2-dimensional tables), and in each simulated image, counting the number of stars in a binary system. This task can be automated and performed in a few minutes, with modern computers, a good random number generator, and a smart algorithm.

It could be slow if you use a naive approach. You can do much better than O(n^2) in terms of computational complexity, to compute the n distances to the nearest stars. The idea is to store the data in a grid where granularity = 1 mm (that is, a 2-dim array with 100 x 100 = 10,000 cells). Thus for each star, you have to look only at the eight surrounding pixels to count the number of neighbors less than 1 mm away. The O(n^2) complexity has been reduced to O(n), at the expense of using 10,000 memory cells of 1 bit each (presence / absence of a star).

Note that I picked up the number 1,000,000 arbitrarily, but in practice it needs to be just big enough so that your estimates are stable enough, with additional simulations bringing few or no corrections. Selecting the right sample and sample size is a experimental-design problem, and using model-free confidence intervals facilitates this task and makes the results robust. This Monte Carlo simulation approach is favored by operations research professionals, as well as by some data scientists, computer scientists, and software engineers who love model-free statistical modeling. However, in this case the theoretical model is well known, simple if not elementary, and comes with a quick, simple answer. So unless you have to spend hours understanding how it works or to just discover its existence, go for the theoretical solution in this case

**Caveat**: In this example, stars are seen through a 2-D screen. But in reality, they lie in a 3-D space. Two stars might appear as neighbors because their X and Y coordinates are close to each other, but could be eons apart on the Z-axis. So to compute the real expected proportion of binary stars, you would have to simulate stars (points) in 3-D, then project them on a rectangle, and then count the binary stars. I'm not sure the theoretical model offers a simple solution in this case, but the Monte Carlo simulations are still straightforward. In practice, stars that are actually far away are not shiny enough to show up on the picture, so the 2-D model is indeed a good approximation of the real 3-D problem.

Also, in the theoretical model, some implicit independence assumptions are made regarding star locations (when mentioning the binomial model), but this is not the case in practice, because the 1 mm circles centered on each star sometimes overlap. The approximation is still good and is conservative — in the sense that the theoretical number, when corrected for overlap, will be even higher than 73.

## Scenario 3: Data Glitch Turns Data Into Gibberish

There are many examples available of data screw-ups occurring during the import or export of data. It is usually the result of poor architecture, using the wrong tools, or blindly relying on them without performing data quality checks. How does a data scientist detect and fix this?

Microsoft Access sometimes cannot properly import or export data types. Automated and even forced type casting from a database table to text or CSV files (or the other way around) is poor, resulting in many errors. This occurs because the environment is not flexible enough to do smart type detection and casting. For example, have you ever dealt with a date that was recorded as an integer in one row and mmddyy (month, day, and year, as in 100516 for October 5, 2016) in the next row? Or perhaps you've had to deal with a comma inside a data field value that screws up your CSV file?

I've received many data files that had a data corruption rate above 5 percent. Sometimes two different types of values (advertiser keyword versus user query, or referral domain versus ad network domain) are stored in the same field. Sometimes the fields are not properly aligned. Sometimes it looks like the people who produced the data were not experts in regular expressions or used separators badly. Sometimes it's the software that messes things up.

Sometimes these screw-ups have undesirable side effects, such as one single artificial user ID in your database becoming a *garbage collector* for all users with no user ID, resulting in flaws when analyzing the data. At other times it's not a big deal.

Here are a few examples where messy data, messy formatting, and broken data integration resulted in serious errors, as well as how the problems were fixed.

- ***Wells Fargo***: Tealeaf users' sessions were broken down into multiple sessions because each server had its own web log and the blending of all the web logs was not done properly. In this case, it was mostly an architecture design issue. Fortunately, the problem was discovered and fixed, and great insights were gained along the way.
- ***eBay***: A data transfer removed special characters from French and German keywords, making reconciliation impossible. A temporary lookup table of foreign keywords was built, with correct and erroneous spellings, to fix the problem.
- ***Click fraud detection***: The keyword field sometimes represented a user query (such as a Google query), and sometimes a pre-specified keyword category, depending on the type of ad network affiliate. Affiliates where the keyword was indeed a keyword category were erroneously heavily penalized because the keyword distribution was (by design, not by fraud) very poor. Adding a new field that specified the type of keyword fixed the problem.

As an exercise, I invite you to think more about these types of issues and answer the following questions. These are also interesting job interview questions.

- How do you address these issues?
- How do you automatically detect these screw-ups? How is it a quality assurance issue?
- Is it worse with big data? Is detecting or fixing it more difficult?
- How much bad data can we tolerate? One percent, or even less in problems such as fraud detection?
- What proportion of your time is wasted on fixing these issues?
- How can you design smart type casting?

## Scenario 4: Regression in Unusual Spaces

This example illustrates the need to adapt old techniques, such as regression, before applying them to new problems. Some of these techniques are more than 100 years old, and have been the workhorses of statistical analysis. They were developed when data sets were small, and simple mathematical solutions could solve problems. This is no longer the case with the advent of big data and massive paralleled computing power.

Say you want to reverse-engineer the recipe for Coca-Cola. The response, Y, is how close the recipe is to the actual Coca-Cola recipe, based on a number of tastings performed by a number of different people (according to the experimental design). Indeed, it's quite similar to a clinical trial where a mix of atoms or chemical radicals (each combination producing a unique molecule) is tested to optimize a drug. The independent variables are binary, each one representing an ingredient, such as salt, water, corn syrup, and so on. The value is equal to 1 if the ingredient in question is present in the recipe, and 0 otherwise. It's quite different from standard linear or logistic regression.

**TECHNICAL NOTE**

**The regression coefficient a_k (k = 1,...,m) must meet the following requirements:**

- **Each k is positive (a_k is greater or equal to 0).**
- **The sum of these coefficients is equal to 1.**

**In short, you're doing a regression on the simplex domain where a_k represents the proportions of a mix. An interesting property of this regression is that the sum of the square of the a_k coefficients is equal to the square of the area of the m-1 dimensional face defined by SUM(a_k) = 1, and a_k is greater than or equal to zero. (This is just a generalization of Pythagoras's theorem.) It is a bit like a lasso, ridge, or logic (not logistic) regression, but it could also be solved using Markov Chain Monte Carlo (MCMC) in a Bayesian framework.**

And what about solving a regression on a sphere? For instance:

- Identify the Canada Goose's migration flight paths based on bird sightings.
- Assess the trajectory and origin of a meteor that resulted in multiple aligned impact craters on a planet.
- Detect tectonic faults based on volcano locations (almost aligned, as in the Cascade Range, or in the mid-Atlantic, under the sea).
- Solve the regression of a plane by mapping a sphere onto the plane.

In this example, the intent is to create a competing product that tastes the same as Coca Cola, call it by a different name, and sell it for far less. If the taste is identical but the ingredients are different, then the Coca Cola manufacturer won't be able to successfully sue you for copying their recipe. I think Virgin almost managed to create a clone. And of course, Pepsi does not come close: the taste is so different, just like apples and oranges.

Finally, there are many different ways to solve this regression problem (or any data science problem. The solutions tend to be equivalent in terms of efficiency if you use the right parameters. For example, you could solve this problem with pure optimization/mathematics. Statisticians would argue that this approach would not allow you to build confidence intervals for the regression parameters, or test whether some are equal to zero. Alternately, a methodology of my own design computes confidence intervals without statistical models. The technique, known as the First Analyticbridge Theorem, is described in the more technical chapters of this book.

## Scenario 5: Analytics Versus Seduction to Boost Sales

This example illustrates that even the best analytics are not very useful if you ignore other critical components that make a business successful. In short, analytics is not a panacea for all business problems.

The context here is about increasing conversion rates, such as converting website visitors into active users, or passive newsletter subscribers into business leads (a user who opens the newsletter and clicks on the links becomes a lead). Here the newsletter conversion problem is discussed, although this example can apply to many different settings.

To maximize the total number of leads, you need to use both seduction and analytics. Sales is a function of:

- Seduction
- Analytics
- Product
- Price

- Competition
- Reputation
- Marketing

**NOTE** **How to assess the weight attached to each of these factors is beyond the scope of this chapter.**

First, even measuring "seduction" or "analytics" is difficult. But you could use a 0-to-10 scale, 10 being the best, with seduction = 9 representing a company using significant efforts to seduce prospects, and analytics = 0 representing a company totally ignoring analytics.

In the context of newsletter optimization (to maximize lead quantity and volume), most companies set seduction to 1 and analytics to 4 or 5. Analytics is usually achieved through multivariate testing and mathematical segmentation and targeting of the user base. This approach originates from the way marketing people are trained — it is not, however, the best approach. Segmentation by ISP rather than by traditional user group is now critical. *Failing* to reach Gmail users is far worse than reaching Gmail users through a poor user segmentation (for example, where young and old users receive different messages, but not with efficient targeting).

Another critical mistake is to use the same keywords in the subject line over and over, which may work well at first, but eventually bores your subscribers to the point that they don't read your newsletter anymore (unless you've found a way to beat churn, maybe by hiring a growth hacker). The problem is compounded if your competitors use exactly the same keywords.

A rich variety of non-hype keywords works well with an analytic, highly educated, spam-averse audience. For example, a subject line such as *Weekly digest, July 29* (with a lowercase *d* in *digest*) works better than *25 Fantastic Articles From Top Big Data Gurus* (with an uppercase *F* in *From*). Sure, the latter will work well maybe two times, but eventually it stops working. In addition, the content of your newsletter must match the subject line; otherwise you will lose subscribers faster than you can acquire new ones.

This contrarian approach is based on Seduction 101 rather than Analytics 101 — that is, guessing what the users like to read today rather than designing content based on historical performance. Maybe it can be automated and turned into analytic rules — for example, by detecting how many times you can use hype, how long a great keyword such as "belly dancing analytics" will work, and so on. Overusing the tricks detected through analytics eventually kills seduction, as well as sales in the process. But for now, many online marketers still seem to ignore these rules. Seduction can provide a bigger lever than analytics (although they should be blended together), especially when building a business for the long term.

## Scenario 6: About Hidden Data

In this scenario, your data is like Gruyere cheese. It has holes. Big holes. Sometimes the empty space occupies a bigger volume than the data itself — just like dark matter is more abundant than visible matter in the universe. This example is not about shallow or sparse data, but instead about data that you do not see, that you do not even know exists, yet contains better actionable nuggets than anything in your data warehouse.

Following are three cases of "Gruyere data," along with the remedy for each case.

### *Missing or Incomplete Data*

This is the easiest problem to fix. Any talented data scientist can work around this issue using modern, unbiased imputation techniques. Most analytic software also includes mechanisms to handle missing data.

### *Censored Data*

By censored, I mean censored from a statistical point of view. Here's an example that comes to mind: You want to estimate the proportion of all guns involved in a crime at least once during their lifetime. The data set that you use (gun or crime statistics) is censored, in the sense that a brand new gun has not killed someone today but might be used in a shooting next week. Also, some criminals get rid of their gun, and it might not be traceable after the crime.

How do you deal with this issue? Again, any talented data scientist will easily handle this problem using a statistical distribution (typically exponential) to measure time-to-crime, and estimate its mean based on censored data, using correct statistical techniques. Problem solved.

### *Hidden Data*

Dealing with hidden data is a big issue. First, you don't even know it exists because it is invisible, at least from your vantage point. Domain expertise and statistical knowledge (rules of thumb more than technical knowledge) help you become aware of potential hidden data. Indeed, the data might not exist at all, in which case you have to assemble the data first.

Consider this example: Let's say Target is trying to optimize its revenue numbers. It analyzes its sales data to see when garden items sell best. It has no data about selling garden stuff in February — the company headquarters are in Minnesota, and anyone suggesting such an idea might be fired on the spot, or suspected of being on drugs. Yet in California, Target's competitors are all selling garden stuff in February, leaving next to nothing for Target, in terms of sales, when June comes around. Target, unaware of the cause, thinks there's not much money to be made on garden items in California.

How do you address this issue? Even though Target may lack data for garden sales in February, you could look at competitor data (for instance, scanning and analyzing the millions of pieces of junk mail sent every day) as a good first step in the right direction. But the real solution is to hire a *visionary data scientist*. Talented data scientists leverage data that everybody sees; *visionary data scientists* leverage data that nobody sees.

## Scenario 7: High Crime Rates Caused by Gasoline Lead. Really?

Here is a typical example of a study you may read in respected news outlets, yet the analytics used to support the author's opinion are poor. Crime rates in big cities (where gasoline use is high) peaked about 20 years after lead was banned from gasoline, according to an econometric study by Rick Nevin. The 20-year time lag is the time elapsed between lead exposure at birth and turning into a 20-year-old criminal. At least that's the argument proposed by some well-known econometricians, based on crime-rate analysis over time in large cities. This is another example of a study done without proper experimental design.

So how would you design a better study? You could get a well-balanced sample of 10,000 people over a 30-year period across all cities of a certain size, split the sample into two subsets (criminals versus non-criminals), and check (using an odds ratio) whether criminals are more likely to have been exposed to lead at birth than non-criminals. In short, do the opposite of the original study and look at individuals rather than cities — that is, look at the micro rather than macro level — and perform a classic test of the hypothesis using standard sampling and proper design of experiment procedures.

Alternatively, if you really want to work on the original macro-level time series (assuming you have monthly granularity), then perform a *Granger causality test*, which takes into account all cross-correlation residuals after transforming the original time series into white noise (similar to spectral analysis of time series, or correlogram analysis). However, if you have thousands of metrics (and thus thousands of time series and multi-million correlations), you will eventually find a very high correlation that is purely accidental. This is known as the "curse" of big data, described in detail in the next chapter.

Correlation is not causation. Don't claim causation unless you can prove it. Many times, multiple inter-dependent factors contribute to a problem. Maybe the peak in crime occurred when baby boomers (a less law-abiding generation) reached the age of 20. This may be a more credible cause.

## Scenario 8: Boeing Dreamliner Problems

The new Dreamliner plane from Boeing was grounded by the FAA for a few months shortly after its launch due to problems related to its batteries. The main problem was a new type of lithium-ion battery that had never been used in a

plane before. This type of powerful battery easily overheats and catches fire, which resulted in a number of emergency landings over a short period of time.

The root cause was the lack of good experimental design by the vendor that designed the batteries. It was a quality control issue, and quality control relies heavily on analytics. The following questions support the fact that better quality control and experimental design could address problems like this one:

- Aren't these batteries (like pretty much any product that you can purchase, such as a car or laptop battery) going through extensive quality control testing, using sound statistical techniques to make sure that faulty batteries, or risk of failure over the lifetime of the product, is below an acceptable threshold?
- Could it be that the quality control tests were not performed according to best practices?
- Were the overheating simulations representative of real-world conditions as found in an airplane taking off?
- Did they not "stress" the battery for long enough?
- Are standards for quality control lower in Japan, which is where the battery was designed and produced?
- Were the statistical reports about the reliability of these batteries incorrect?

A possible solution is the use of better mechanisms to cool this type of battery, which had never been used in an airplane before but is found in all cell phones and was responsible for some spectacular cell phone fires in the past. Unlike in a cell phone or a laptop, it seems easy to cool (and even freeze) anything in a plane because the outside temperature is well below the freezing point.

## Scenario 9: Seven Tricky Sentences for NLP

NLP means *Natural Language Processing*. The kind of problems that a data scientist faces when analyzing unstructured data, such as *raw* (uncategorized) *text*, is illustrated here. This type of analysis is known as *text mining*.

**CROSS-REFERENCE** **Chapter 2 discusses how to create a structure (typically a taxonomy) on unstructured data.**

Following are seven types of language patterns that are difficult to analyze with automated algorithms:

- "A land of milk and honey" becomes "A land of Milken Honey" (algorithm trained on *The Wall Street Journal* from the 1980s, where Michael Milken was mentioned much more than milk).
- "She threw up her dinner" versus "She threw up her hands."

- "I ate a tomato with salt" versus "I ate a tomato with my mother" or "I ate a tomato with a fork."
- Words ending with "ing" — for instance, "They were entertaining people."
- "He washed and dried the dishes," versus "He drank and smoked cigars" (in the latter case he did not drink the cigars).
- "The lamb was ready to eat" versus "Was the lamb hungry and wanting some grass?"
- Words with multiple meanings, such as "bay" to refer to a color, type of window, or body of water.

In these examples, as well as in sentiment analysis, the data scientist is trying to guess the user intention in order to correctly interpret the data and provide the right type of answer or decision. For instance, this could happen in the following situations:

- When analyzing Google search data, which relevant ads should you display based on the user query?
- When analyzing comments posted on a Facebook page about a product or company, how do you assess whether they are positive or negative?
- When analyzing Facebook posts, how do you decide if a post should be automatically deleted because it violates Facebook policies or some laws?

Using proxy metrics based on user profiles (if available), or web pages and e-mail where the content is found, can help resolve ambiguities, especially if the web pages are already categorized.

## Scenario 10: Data Scientists Dictate What We Eat?

There are many factors that influence what the average American eats, but the biggest factor is the margin the grocery store makes on the products it sells. This explains why you can't get red currants or passion fruit anymore, but you can find plenty of high-energy drinks and foods rich in sugar. Of course, there's a feedback loop: Americans like sweet stuff, so many companies produce sweet food, and due to large-scale processing it's cheap, can be priced efficiently by grocery stores, and sells well.

Here's how a supermarket could increase revenue with better analytics. Behind all of this is data science, which helps answer the following questions:

- Which new products should be tested for customer interest and return on investement (ROI)? Red currant pie? Orange wine? French-style cherry pie? Wild boar meat? Purple cheese? Red eggs? Cheese shaped like a ball? (Although anything that is not shaped like a parallel-piped rectangle is suboptimal from a storage point of view, but that's another data science issue.)

- How do you determine success or failure for a new product? How do you test a new product (experimental-design issue)?
- Which products should be eliminated? (Passion fruit, passionfruit juice, and authentic Italian salamis are no longer sold in most grocery store in the United States.)
- How do you measure lift (increased revenue)? Do you factor in costs of marketing and other expenses?
- How do you price an item?
- How do you cross-sell?
- How do you optimize ROI on marketing campaigns?
- When and where do you sell each product (seasonal and local trends)?
- How do you improve inventory forecasting?

The last time I went to a grocery store, I wanted to buy full-fat unsweetened yogurt. It took me 10 minutes to find the only container left in the store (the brand was Dannon). I was ready to pay more to get that particular yogurt (a product that has been consumed worldwide by billions of people over several millennia) rather than the two alternatives: low fat, or plain but sweetened. (Ironically, the "low fat" version had 180 calories per serving, whereas the old-fashioned plain yogurt had 150. This is because they add corn starch to the low-fat product.)

Over time, I've seen the number of grocery product offerings shrink. More old products are eliminated than new products are introduced. Clearly the products eliminated are those with a smaller market, such as passion fruit. But could data science do a better job helping grocery retailers decide what goes on the shelves, when and where, in what proportions, and at what price?

The answer is yes. A good solution would be the use of more granular segmentation with lower variance in forecasted sales and revenue (per product) due to use of models with higher predictive power. In the case of the yogurt, while a broad spectrum of people try to avoid fat, there are plenty of thin people on the west and east coasts who don't mind eating plain yogurt. So it could make sense to sell plain yogurt in Seattle or Boston (maybe just a few containers with a high price tag, among dozens of cheaper low-fat brands) but not in Kansas City.

This would also create new opportunities for specialized grocery stores, like PCC Natural Markets in the northwest United States, selling precisely what other supermarkets have stopped selling (as long as it is sellable). In short, selling stuff that generates profit but that other supermarkets have written off.

This example also shows how communication skills are important for data scientists: to propose a new approach and convince senior management that there is a way to simultaneously optimize profit and bring long-term value to the customers. Of course, such an approach would be a long-term strategic investment,

and the supermarket may not able to meet the financial numbers in the short term (which is something the CEO will need to address with shareholders).

## Scenario 11: Increasing Amazon.com Sales with Better Relevancy

How could Amazon.com increase sales by redefining relevancy? Answer: By improving its search and relevancy engines to include item price as a main factor. The type of optimization and ROI boosting described here applies to all digital catalogs, although we are focusing on books in this discussion.

### *Search Engine*

When you perform a keyword search on Amazon.com in the book section, it returns a search result page with, say, 10 suggested books matching your keywords. This task is performed by the search engine. The search engine displays the books in some type of order. The order is based either on price or keyword proximity.

### *Relevancy Engine*

If you search for a specific book title, Amazon.com also displays other books that you might be interested in based on historical sales from other users. This task is performed by the relevancy engine.

**TECHNICAL NOTE**

**The relevancy engine works like this: if m(A,B) users both purchased book A (the book you want to purchase) and another book B over the last 30 days, and if k(A) users purchased A, and k(B) users purchased B, then the association between A and B (that is, how closely these books are related from a cross-selling point of view) is defined as R(A,B) = m(A,B) / SQRT{k(A) * k(B)}. The order in which the suggested books are displayed is entirely determined by the function R(A,*).**

### *Better Sorting Criteria*

Expensive books generate very few sales, but each sale generates huge profit. Cheap books generate little money, but the sales volume more than compensates for the little profit per book. In short, if you show books that all have exactly the same relevancy score to the user, the book that you should show in the #1 position is the book with the optimum price with regard to total expected revenue. Figure 1-1 shows a hypothetical optimum book price of $21.

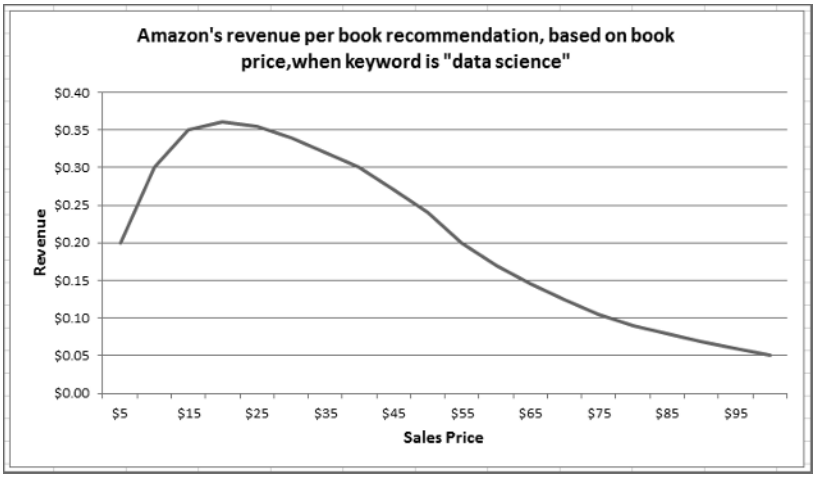


**Figure 1-1:** Optimum book pricing

This chart is based on simulated numbers, assuming that the chance for a sale is an exponentially decreasing function of the book price. That is:

$$P(\text{sale} \mid \text{price}) = a * \exp(-b*\text{price})$$

A more general model would be:

$$P(\text{sale} \mid \text{price, relevancy score}) = a * \exp(-b*\text{price}) * f(\text{relevancy score})$$

Another way to further increase revenue is to include user data in the formula. Some users have no problem purchasing an expensive book. Users who traditionally buy more expensive books should be shown more expensive books in their search results.

### *Putting It All Together*

When a sale takes place, how do you know if it is because of showing rightly priced books at the top, or because of perfect relevancy? For instance, relevancy between "data science" and "big data" is good, but relevancy between "data science" and "cloud computing" is not as good. Does it make sense to suggest an expensive "cloud computing" book to a wealthy user interested in a "data science" book, or is it better to suggest a less expensive book related to "big data" if your goal is to maximize profit? It also depends on how you define revenue optimization: is it long term (then relevancy is more important) or short term? Usually it's a blend of short term and long term. As you can see, separating the influence of relevancy from the price factor is not easy.

The price factor is particularly useful when keyword or category relevancy is based on "small data," which is a specialized user query or a book with little volume. Also, detecting what causes a specific conversion or sale is a complex

problem, known as *attribution*. In sophisticated settings, macro-economic (long-term, aggregate) indicators are used in the context of *marketing mix optimization*, and are blended with highly granular real-time attribution metrics. Price is also analyzed using *price elasticity models* and *efficiency curves*.

Another area of interest is customized pricing, where the price of a book is determined in real time, based on the user, sales history of the user (if available), the website where the book is sold (based on website demographics), and the book itself. Some studies suggest that a fixed price works best, as savvy users would try to purchase the same item multiple times until they are offered a price as low as possible (though they have no way to know when the minimum price is offered). Yet, selling at a higher price to users who don't mind purchasing expensive books (based on their sales history), and not recommending a book the user has already purchased, are low-hanging fruits for ROI improvement. However, such pricing strategies have been banned in some countries, and are controversial elsewhere. Using price customization, especially if secretly carried out without user consent and knowledge, via unethical data mining practices and privacy violations, could cost you the trust of your users, and eventually result in churn, lawsuits, and revenue drop. Eventually, users will notice that prices are customized.

Another way to maximize Amazon.com's profit would be to print books *on demand* rather than managing an inventory and having to forecast sales for each book and each month. Data science also helps determine where warehouses should be located to optimize delivery (in terms of speed and cost), as well as sales tax optimization to better compete with other booksellers.

## Scenario 12: Detecting Fake Profiles or Likes on Facebook

Some websites sell fake Twitter followers ($10 for 1,000 followers), fake Facebook accounts, or even fake Yahoo! mail accounts to help spread spam or make the person or retailer appear more popular than it really is, supposedly to boost sales. They tell the buyer up front that these followers are fakes. They also sell Facebook Likes, though they claim that they are from "real" Facebook profiles.

The easiest way to automatically detect fake Likes is by looking at the number of relevant comments: if a Facebook (or any) post has 5,000 Likes and either no comments or 20 comments that are just variations of "this is a great post," then you can bet the Likes and comments are mass-produced by a robot or an army of zombies — they are fake.

Data science algorithms that rely on such metrics as well as on social graphs, velocity, and recency in account creation and postings, natural language processing, and Botnet/fraud detection rules, are used to handling this type of problem. Some fakes can be detected in real time, while some can be detected and eliminated later, such as after ad hoc analyses (reactive analytics performed by data analysts, after the fact) or automatically by end-of-day algorithms (prescriptive analytics).

### Scenario 13: Analytics for Restaurants

Analytics can take various forms for small businesses, and restaurants in particular. Some of the problems these businesses need to solve include pricing optimization, inventory management, sales forecasting, access to competitor information and what your clients say about you (you need to regularly check restaurant reviews with some automated software), space optimization (how many tables you can squeeze in), wine and cheese selection, frequency in menu changes, hours and days when should you open and close, number of dishes that you offer, and how you keep your chef (including deciding on optimum salary.)

Possibly one of the easiest problems to solve is the table layout. The optimum solution consists of having your two-seat tables stacked against the wall, and your four-seat tables in the center of the room. Should these tables be round or square, parallel to the wall, or forming a 45-degree angle? It's easy to find a mathematical solution, or at least do some simulations to find out what works best. Square tables are better than round ones, and they allow you to easily combine them to accommodate larger parties. If you increase seating capacity by 20 percent, then your profit should also increase by 20 percent, assuming that you are always fully booked and profitable.

## Data Science History, Pioneers, and Modern Trends

As data science has grown, other related professions have seen a decline, such as statistics and computer science. Publicly available data from Google (see `http://bit.ly/1aF8D5T`) shows the following:

- An increase in the number of data analysts since 2010
- A decline in the number of statisticians since 2005
- A decline in the number of computer scientists since 2005
- An explosion in the number of data scientists since 2012

You can find other public data on LinkedIn (number of applicants per job ad) or Indeed, but they tend to be job-market related.

Other similar data show that all traditional fields are declining: six sigma, data mining, statistics, and operations research. Big data started to emerge and grow exponentially around 2011, according to Google, and by 2013 was more popular than data mining or six sigma. Even though the rise of big data is dramatic, the explosion of searches for the keyword *analytics* is even more spectacular and started in 2006, according to the same source. It tapered off in 2012 at a level six times higher than *big data*.

Of course, many professionals (including me) who, in the year 2000, were doing statistics, operations research, computer science, six sigma, actuarial science, biostatistics, or some other narrowly defined analytic activity, have gained

experience, leadership, broader knowledge, business acumen, and expertise spanning many fields. Thus they changed their job titles, but they all share something in common: analytics. At the same time, the growth of data and modern data architecture, such as MapReduce, hitting all disciplines has acted as a common denominator and cement for many related professions.

**NOTE** ***Data scientist*** **is broader than** ***data miner*****, and encompasses data integration, data gathering, data visualization (including dashboards), and data architecture.** ***Data scientist*** **also measures ROI on data science activities.**

## Statistics Will Experience a Renaissance

A lot has been said about the death of statistics, some of it by leading statisticians themselves. I believe statistical science will eventually come back, but it will be more applied, adapted to big data, and less model-driven. It will be integrated with computer science, predictive modeling, data mining, machine learning, some aspects of operations research and six sigma, and database architecture, under a big umbrella called data science, business analytics, decision science, data intelligence, analytics, or some other term yet to be created or reused. We are currently in the middle of this analytics revolution.

In particular, guys like me, although having a new job title — data scientist — still do statistics part-time, and even theoretical cutting edge statistics at times. In my case, I am reviving a quarter of a century-old but robust technique that was deemed too complicated in 1750 due to lack of computing power, and was abandoned. The lack of computing power in 1750 resulted in new, mathematically friendly techniques developed around 1800, with simple formulas such as least square regression. This framework has survived to this day and could be the cause of the current decline of traditional statisticians because robustness is more important than ever with big data, and computational complexity is no longer an issue when gigabytes of data can be processed in a few minutes on distributed systems (in the cloud with MapReduce). Also, most modern scientists, geographers, physicians, econometricians, operations research professionals, engineers, and so on have a decent, applied statistical knowledge. However, software engineers and computer scientists sometimes ignore or misuse statistical science, sometimes as badly as journalists, with bad consequences such as development of systems (for example, recommendation engines) with large numbers of undetected fake reviews and fraud. Eventually, statistical science will start pouring into these communities.

Some people say that most data analysts are statisticians. In my opinion, data analyst is a junior title, typically a person with a BS or BA degree. Statisticians have a more theoretical background, are trained to use models developed before

the advent of big data, and have an MS or PhD degree. People spending their days writing SQL queries and reports are data analysts.

Part of the reason I moved away from being called a statistician is because of the American Statistical Association: it changed the meaning of the keyword *statistician*, as well as limited career prospects for future statisticians by making it almost narrowly and exclusively associated with the pharmaceutical industry, government (surveys, census, politics), and small data (from where most of its revenue derives). The Association has generally stayed outside the new statistical revolution that has come along with big data over the last 15 years. As a Belgian citizen, I can say the same about the Belgian Statistical Association. So this trend is not limited to the United States, and it is not limited to the (American) English language, but also includes the French and Dutch languages, among others.

Statisticians should be very familiar with computer science, big data, and software — 10 billion rows with 10,000 variables should not scare a true statistician. On the cloud (or even on a laptop as streaming data), this amount of data gets processed quickly. The first step is data reduction, but even if you must keep all observations and variables, it's still processable. A good computer scientist also produces confidence intervals: you don't need to be a statistician for that, just use the First Analyticbridge Theorem discussed later in this book. The distinction between computer scientist and statistician is getting thinner and fuzzier. No worries, though — the things you did not learn in school (in statistical classes), you can still learn online.

## History and Pioneers

Now let's look at the history of data science and companies who were pioneers in analytics and data science. First, take a look at popular keywords used since the late 1980s, and one prediction for 2022:

- 1988: artificial intelligence. Also: computational statistics, data analysis, pattern recognition, and rule systems.
- 1995: web analytics. Also: machine learning, business intelligence, data mining, ROI, distributed architecture, data architecture, quant, decision science, knowledge management, and information science.
- 2003: business analytics. Also: text mining, unstructured data, semantic web, Natural Language Processing (NLP), Key Performance Indicator (KPI), predictive modeling, cloud computing, lift, yield, NoSQL, Business Intelligence (BI), real-time analytics, collaborative filtering, recommendation engines, and mobile analytics.
- 2012: data science. Also: big data, analytics, software as a service (SaaS), on-demand analytics, digital analytics, Hadoop, NewSQL, in-memory analytics, machine-to-machine, sensor data, healthcare analytics, utilities analytics, data governance, and in-column databases.

- 2022: data engineering. Also: analytics engineering, data management, data shaping, art of optimization, optimization science, optimization engineering, business optimization, and data intelligence.

Each of these milestones brings us to a more generic, global, comprehensive understanding of leveraging data. Google was one of the first significant contributors to the big data movement, starting around 1995. Google solved the database storage capacity limitations associated with traditional distributed systems/ Database Management System (DBMS) systems by introducing the Google file system, MapReduce. (Big tables were often discussed in industry conferences from 2003 to 2006.) Then came HBase and Hadoop Distributed File System (HDFS). In addition to Google, Yahoo!, and Facebook have also made significant contributions in the Hadoop and open source community that drives technology advancement.

Regarding the pioneering companies in the purely analytic sphere, consider the following 26 and their respective Wikipedia pages where you can learn a lot of interesting history. (Warning: Some of these web pages, as on any Wikipedia topic, contain commercial content promoted by people with conflicts of interest, insiders, or people with interests in the companies in question).

- Datawatch (`http://en.wikipedia.org/wiki/Monarch`)
- Excel (`http://en.wikipedia.org/wiki/Microsoft _ Excel`)
- FICO (`http://en.wikipedia.org/wiki/FICO`)
- Greenplum (`http://en.wikipedia.org/wiki/Greenplum`)
- IMSL (`http://en.wikipedia.org/wiki/IMSL _ Numerical _ Libraries`)
- Informatica (`http://en.wikipedia.org/wiki/Informatica`)
- KNIME (`http://en.wikipedia.org/wiki/KNIME`)
- KXEN (`http://en.wikipedia.org/wiki/KXEN _ Inc.`)
- Lavastorm (`http://en.wikipedia.org/wiki/Lavastorm`)
- MapReduce (`http://en.wikipedia.org/wiki/MapReduce`)
- Mathematica (`http://en.wikipedia.org/wiki/Mathematica`)
- Matlab (`http://en.wikipedia.org/wiki/Matlab`)
- Netezza (`http://en.wikipedia.org/wiki/Netezza`)
- NoSQL (`http://en.wikipedia.org/wiki/NoSQL`)
- Oracle (`http://en.wikipedia.org/wiki/Oracle _ Database`)
- PMML (`http://en.wikipedia.org/wiki/Predictive _ Model _ Markup _ Language`)
- R programming language (`http://en.wikipedia.org/wiki/R _ `)
- RapidMiner (`http://en.wikipedia.org/wiki/RapidMiner`)
- S-PLUS (TIBCO) (`http://en.wikipedia.org/wiki/S-PLUS`)

- SAS (http://en.wikipedia.org/wiki/SAS _ Institute)
- Splunk (http://en.wikipedia.org/wiki/Splunk)
- SPSS (http://en.wikipedia.org/wiki/SPSS)
- Statistica (http://en.wikipedia.org/wiki/STATISTICA)
- Sybase (http://en.wikipedia.org/wiki/Sybase)
- Tableau (http://en.wikipedia.org/wiki/Tableau _ Software)
- Teradata (http://en.wikipedia.org/wiki/Teradata)

This list focuses on large analytics companies that have been around for some time. Countless others have sprung up in the last few years, especially around the MapReduce ecosystem.

**CROSS-REFERENCE** **More information about the big data ecosystem is found in Chapter 2.**

## Modern Trends

In terms of new technology, much about what is going on today revolves around integrating analytics in big distributed databases. It is about having the data architect or engineer and the data scientist communicate with each other, moving away from the old dichotomy — data-to-analytics (the traditional data science approach) versus analytics-to-data (the more modern approached favored by data architects and database companies because it is faster). Analytics-to-data means performing the analytics inside or close to where the database or data environment resides, rather than downloading data locally and then processing it. This eliminates a lot of redundant downloads by multiple database users. Read the last section in this chapter for more details.

This, of course, boils down to building the right advanced analytics tools (not just the Extract, Transform, Load (ETL) aspects) in or close to where the database resides. When analytics is inside the database, it is sometimes called in-memory analytics. It is a stronger form of analytics-to-data in which analytics is integrated and embedded into the database architecture and takes place mostly in fast memory (RAM), sometimes using multiple servers. One of the issues is that modern database processing involves more complex programming languages, and right now most people are still using SQL. It is hard to change an old habit. So pioneering companies such as Pivotal have come up with a system called Fast SQL, where programmers accustomed to SQL don't need to learn a new more complicated language, and where the code is optimized to run under Hadoop (a distributed architecture).

Other modern trends include automated machine-to-machine communications in real time, such as in high-frequency trading strategies or massive

bidding systems. An example of this is eBay updating keyword bids for 10 million keywords on Google pay-per-click campaigns, every day and in real time, based on keywords' past performance (when history is available) or analytic scoring for new keywords with no history. Some of these machine-to-machine communications are done via APIs or AaaS (on-demand analytics as a service). An API call is nothing more than an HTTP request (actually, an HTTPS request most of the time), with parameters (key/value pairs encoded in XML format) in the query string used to specify the type of services needed (such as keyword price and volume forecasts).

Also, what makes modern data different and unique is its variety (sometimes coming in unstructured format from Twitter tweets, well structured from mobile devices, or from sensors), its velocity, and its volume, which makes traditional statistical analyses not always suitable.

To summarize, these are the characteristics of the modern trends in data science:

- In-memory analytics
- MapReduce and Hadoop
- NoSQL, NewSQL, and graph databases
- Python and R
- Data integration: blending unstructured and structured data (such as data storage and security, privacy issues when collecting data, and data compliance)
- Visualization
- Analytics as a Service, abbreviated as AaaS
- Text categorization/tagging and taxonomies to facilitate extraction of insights from raw text and to put some structure on unstructured data

A final word about Perl: This great programming language, still popular five years ago, is a casualty of modern programming environments. It has been replaced by Python and its analytical and graphical libraries. Although very flexible, allowing you to code without concentrating on the code, Perl programs were very hard and costly to maintain. Perl did not survive in modern agile and lean environments. Some have said that Excel — the most popular analytic tool — would also die. I don't think so. Modern versions of Excel use the cloud, and retrieve data from the Internet and store it in cubes.

## Recent Q&A Discussions

I recently had the following discussions with a number of data architects, in different communities — in particular with (but not limited to) the The Data Warehousing Institute (TDWI) group on LinkedIn. They show some of the challenges that still need to be addressed before this new analytics revolution is

complete. Following are several questions asked by data architects and database administrators, and my answers. The discussion is about optimizing joins in SQL queries, or just moving away from SQL altogether. Several modern databases now offer many of the features discussed here, including hash table joins and fine-tuning of the query optimizer by end users. The discussion illustrates the conflicts between data scientists, data architects, and business analysts. It also touches on many innovative concepts.

**Question:** You say that one of the bottlenecks with SQL is users writing queries with (say) three joins, when these queries could be split into two queries each with two joins. Can you elaborate?

**Answer:** Typically, the way I write SQL code is to embed it into a programming language such as Python, and store all lookup tables that I need as hash tables in memory. So I rarely have to do a join, and when I do, it's just two tables at most.

In some (rare) cases in which lookup tables were too big to fit in memory, I used sampling methods and worked with subsets and aggregation rules. A typical example is when a field in your data set (web log files) is a user agent (browser, abbreviated as UA). You have more unique UAs than can fit in memory, but as long as you keep the 10 million most popular, and aggregate the 200 million rare UAs into a few million categories (based on UA string), you get good results in most applications.

Being an algorithm expert (not an SQL expert), it takes me a couple minutes to do an efficient four-table join via hash tables in Python (using my own script templates). Most of what I do is advanced analytics, not database aggregation: advanced algorithms, but simple to code in Python, such as hidden decision trees. Anyway, my point here is more about non-expert SQL users such as business analysts: is it easier or more effective to train them to write better SQL code including sophisticated joins, or to train them to learn Python and blend it with SQL code?

To be more specific, what I have in mind is a system where you have to download the lookup tables not very often (maybe once a week) and access the main (fact) table more frequently. If you must reupload the lookup tables very frequently, then the Python approach loses its efficiency, and you make your colleagues unhappy because of your frequent downloads that slow down the whole system.

**Question:** People like you (running Python or Perl scripts to access databases) are a Database Administrator's (DBA) worst nightmare. Don't you think you are a source of problems for DBAs?

**Answer:** Because I'm much better at Python and Perl than SQL, my Python or Perl code is bug-free, easy-to-read, easy-to-maintain, optimized, robust, and reusable. If I coded everything in SQL, it would be much less efficient. Most of what I do is algorithms and analytics (machine learning stuff), not querying databases. I only occasionally download lookup tables onto my local machine (saved as hash tables and stored as text files), since most don't change that much from week to week. When I need to update them, I just extract the new rows

that have been added since my last update (based on timestamp). And I do some tests before running an intensive SQL script to get an idea of how much time and resources it will consume, and to see whether I can do better. I am an SQL user, just like any statistician or business analyst, not an SQL developer.

But I agree we need to find a balance to minimize data transfers and processes, possibly by having better analytic tools available where the data resides. At a minimum, we need the ability to easily deploy Python code there in non-production tables and systems, and be allocated a decent amount of disk space (maybe 200 GB) and memory (at least several GB).

**Question:** What are your coding preferences?

**Answer:** Some people feel more comfortable using a scripting language rather than SQL. SQL can be perceived as less flexible and prone to errors, producing wrong output without anyone noticing due to a bug in the joins.

You can write simple Perl code, which is easy to read and maintain. Perl enables you to focus on the algorithms rather than the code and syntax. Unfortunately, many Perl programmers write obscure code, which creates a bad reputation for Perl (code maintenance and portability). But this does not have to be the case.

You can break down a complex join into several smaller joins using multiple SQL statements and views. You would assume that the database engine would digest your not-so-efficient SQL code and turn it into something much more efficient. At least you can test this approach and see if it works as fast as one single complex query with many joins. Breaking down multiple joins into several simple statements allows business analysts to write simple SQL code, which is easy for fellow programmers to reuse or maintain.

It would be interesting to see some software that automatically corrects SQL syntax errors (not SQL logical errors). It would save lots of time for many non-expert SQL coders like me, as the same typos that typically occur over and over could be automatically fixed. In the meanwhile, you can use GUIs to produce decent SQL code, using tools provided by most database vendors or open source, such as Toad for Oracle.

**Question:** Why do you claim that these built-in SQL optimizers are usually black-box technology for end users? Do you think parameters can't be fine-tuned by the end user?

**Answer:** I always like to have a bit of control over what I do, though not necessarily a whole lot. For instance, I'm satisfied with the way Perl handles hash tables and memory allocation. I'd rather use the Perl black-box memory allocation/hash table management system than create it myself from scratch in C or, even worse, write a compiler. I'm just a bit worried with black-box optimization — I've seen the damage created by non-expert users who recklessly used black-box statistical software. I'd feel more comfortable if I had at least a bit of control, even as simple as sending an email to the DBA, having her look at my concern or issue, and having her help improve my queries, maybe fine-tuning the optimizer, if deemed necessary and beneficial for the organization and to other users.

**Question:** Don't you think your approach is 20 years old?

**Answer:** The results are more important than the approach, as long as the process is reproducible. If I can beat my competitors (or help my clients do so) with whatever tools I use, as one would say, "if it ain't broke, don't fix it." Sometimes I use APIs (for example, Google APIs), sometimes I use external data collected with a web crawler, sometimes Excel or cubes are good enough, and sometimes vision combined with analytic acumen and intuition (without using any data) works well. Sometimes I use statistical models, and other times a very modern architecture is needed. Many times, I use a combination of many of these.

**CROSS-REFERENCE** **I have several examples of "light analytics" doing better than sophisticated architectures, including a few in the context of email marketing optimization and stock trading. These are discussed in Chapter 6.**

**Question:** Why did you ask whether your data-to-analytic approach makes sense?

**Answer:** The reason I asked the question is because something has been bothering me, based on not-so-old observations (three to four years old) in which the practices that I mention are well entrenched in the analytic community (by analytic, I mean machine learning, statistics, and data mining, not ETL). It is also an attempt to see if it's possible to build a better bridge between two very different communities: data scientists and data architects. Database builders often (but not always) need the data scientist to bring insights and value out of organized data. And the data scientists often (but not always) need the data architect to build great, fast, efficient data processing systems so they can better focus on analytics.

**Question:** So you are essentially maintaining a cache system with regular, small updates to a local copy of the lookup tables. Two users like you doing the same thing would end up with two different copies after some time. How do you handle that?

**Answer:** You are correct that two users having two different copies (cache) of lookup tables causes problems. Although in my case, I tend to share my cache with other people, so it's not like five people working on five different versions of the lookup tables. Although I am a senior data scientist, I am also a designer/architect, but not a DB designer/architect, so I tend to have my own local architecture that I share with a team. Sometimes my architecture is stored in a local small DB and occasionally on the production databases, but many times as organized flat files or hash tables stored on local drives, or somewhere in the cloud outside the DB servers, though usually not very far if the data is big. Many times, my important "tables" are summarized extracts — either simple aggregates that are easy to produce with pure SQL, or sophisticated ones such as transaction scores (by client, day, merchant, or more granular) produced by algorithms too complex to be efficiently coded in SQL.

The benefit of my "caching" system is to minimize time-consuming data transfers that penalize everybody in the system. The drawback is that I need to maintain it, and essentially, I am replicating a process already in place in the database system itself.

Finally, for a statistician working on data that is almost correct (not the most recent version of the lookup table, but rather data stored in this "cache" system and updated rather infrequently), or working on samples, this is not an issue. Usually the collected data is an approximation of a signal we try to capture and measure — it is always messy. The same can be said about predictive models, the ROI extracted from a very decent data set (my "cache"), the exact original, most recent version of the data set, or a version where 5 percent of noise is artificially injected into it — it is pretty much the same in most circumstances.

**Question:** Can you comment on code maintenance and readability?

**Answer:** Consider the issue of code maintenance when someone writing obscure SQL leaves the company — or worse, when SQL is ported to a different environment (not SQL) — and it's a nightmare for the new programmers to understand the original code. If easy-to-read SQL (maybe with more statements, fewer elaborate high-dimensional joins) runs just as fast as one complex statement because of the internal user-transparent query optimizer, why not use the easy-to-read code instead? After all, the optimizer is supposed to make both approaches equivalent, right? In other words, if two pieces of code (one short and obscure; one longer and easy to read, maintain, and port) have the same efficiency because they are essentially turned into the same pseudocode by the optimizer, I would favor the longer version that takes less time to write, debug, maintain, and so on.

There might be a market for a product that turns ugly, obscure, yet efficient code into nice, easy-to-read SQL — an "SQL beautifier." It would be useful when migrating code to a different platform. Although this already exists to some extent, you can easily visualize any query or sets of queries in all DB systems with diagrams. The SQL beautifier would be in some ways similar to a program that translates Assembler into C++ — in short, a reverse compiler or interpreter.

## Summary

This chapter began with a discussion of what data science *is not*, including how traditional degrees will have to adapt as business and government evolve. The changes necessary involve new ways of processing data, as well as new types of data, including big, unstructured, and high velocity streaming data, sometimes generated by sensors or transaction data.

Then a number of real-life scenarios were considered where data science can be used, and indeed help, in different ways. Finally, you considered the history of data science, including pioneering companies and modern trends.

Chapter 2 considers what makes data science a new discipline. It illustrates how big data is different from data seen even just 10 years ago, why standard statistical techniques fail when blindly applied to such data, and what the issues are and how they should be investigated and addressed.