O'REILLY®

Second Edition

Practical Statistics for Data Scientists

50+ Essential Concepts Using R and Python

Early Release

RAW & UNEDITED

Peter Bruce, Andrew Bruce & Peter Gedeck

Practical Statistics for Data Scientists

2ND EDITION

50+ Essential Concepts Using R and Python

Peter Bruce, Andrew Bruce, and Peter Gedeck

Practical Statistics for Data Scientists

by Peter Bruce, Andrew Bruce, and Peter Gedeck

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (http://oreilly.com). For more information, contact our corporate/institutional sales department: 800-998-9938 or *corporate@oreilly.com*.

Editor: Nicole Tachet

Production Editor: Kristen Brown

Copyeditor:

Proofreader:

Indexer:

Interior Designer: David Futato

Cover Designer: Karen Montgomery

Illustrator: Rebecca Demarest

May 2017: First Edition

May 2020: Second Edition

Revision History for the Early Release

- 2019-10-14: First Release

See http://oreilly.com/catalog/errata.csp?isbn=9781492072942 for release details.

978-1-492-07294-2

[LSI]

Chapter 1. Exploratory Data Analysis

As a discipline, statistics has mostly developed in the past century. Probability theory—the mathematical foundation for statistics—was developed in the 17th to 19th centuries based on work by Thomas Bayes, Pierre-Simon Laplace, and Carl Gauss. In contrast to the purely theoretical nature of probability, statistics is an applied science concerned with analysis and modeling of data. Modern statistics as a rigorous scientific discipline traces its roots back to the late 1800s and Francis Galton and Karl Pearson. R. A. Fisher, in the early 20th century, was a leading pioneer of modern statistics, introducing key ideas of *experimental design* and *maximum likelihood estimation*. These and many other statistical concepts live largely in the recesses of data science. The main goal of this book is to help illuminate these concepts and clarify their importance—or lack thereof—in the context of data science and big data.

This chapter focuses on the first step in any data science project: exploring the data. *Exploratory data analysis*, or *EDA*, is a comparatively new area of statistics. Classical statistics focused almost exclusively on *inference*, a sometimes complex set of procedures for drawing conclusions about large populations based on small samples. In 1962, John W. Tukey (Figure 1-1) called for a reformation of statistics in his seminal paper “The Future of Data Analysis” [Tukey-1962]. He proposed a new scientific discipline called *data analysis* that included statistical inference as just one component. Tukey forged links to the

engineering and computer science communities (he coined the terms *bit*, short for binary digit, and *software*), and his original tenets are surprisingly durable and form part of the foundation for data science. The field of exploratory data analysis was established with Tukey's 1977 now-classic book *Exploratory Data Analysis* [Tukey-1977].

Figure 1-1. John Tukey, the eminent statistician whose ideas developed over 50 years ago form the foundation of data science.

With the ready availability of computing power and expressive data analysis software, exploratory data analysis has evolved well beyond its original scope. Key drivers of this discipline have been the rapid development of new technology, access to more and bigger data, and the greater use of quantitative analysis in a variety of disciplines. David Donoho, professor of statistics at Stanford University and former undergraduate student of Tukey's, authored an excellent article based on his presentation at the Tukey Centennial workshop in Princeton, New Jersey [Donoho-2015]. Donoho traces the genesis of data science back to Tukey's pioneering work in data analysis.

Elements of Structured Data

Data comes from many sources: sensor measurements, events, text, images, and videos. The *Internet of Things* (IoT) is spewing out streams of information. Much of this data is unstructured: images are a collection of pixels with each pixel containing RGB (red, green, blue) color information. Texts are sequences of words and nonword characters, often organized by sections, subsections, and so on. Clickstreams are sequences of actions by a user interacting with an app or web page. In fact, a major challenge of data science is to harness this torrent of raw data into actionable information. To apply the statistical concepts covered in this book, unstructured raw data must be processed and manipulated into a structured form—as it might emerge from a relational database—or be collected for a study.

KEY TERMS FOR DATA TYPES

Continuous

Data that can take on any value in an interval.

Synonyms

interval, float, numeric

Discrete

Data that can take on only integer values, such as counts.

Synonyms

integer, count

Categorical

Data that can take on only a specific set of values representing a set of possible categories.

Synonyms

enums, enumerated, factors, nominal, polychotomous

Binary

A special case of categorical data with just two categories of values (0/1, true/false).

Synonyms

dichotomous, logical, indicator, boolean

Ordinal

Categorical data that has an explicit ordering.

Synonyms

ordered factor

There are two basic types of structured data: numeric and categorical. Numeric data comes in two forms: *continuous*, such as wind speed or time duration, and *discrete*, such as the count of the occurrence of an event. *Categorical* data takes only a fixed set of values, such as a type of TV screen (plasma, LCD, LED, etc.) or a state name (Alabama, Alaska, etc.). *Binary* data is an important special case of categorical data that takes on only one of two values, such as 0/1, yes/no, or true/false.

Another useful type of categorical data is *ordinal* data in which the categories are ordered; an example of this is a numerical rating (1, 2, 3, 4, or 5).

Why do we bother with a taxonomy of data types? It turns out that for the purposes of data analysis and predictive modeling, the data type is important to help determine the type of visual display, data analysis, or statistical model. In fact, data science software, such as *R* and *Python*, uses these data types to improve computational performance. More important, the data type for a variable determines how software will handle computations for that variable.

Software engineers and database programmers may wonder why we even need the notion of *categorical* and *ordinal* data for analytics. After all, categories are merely a collection of text (or numeric) values, and the underlying database automatically handles the internal representation. However, explicit identification of data as categorical, as distinct from text, does offer some advantages:

- Knowing that data is categorical can act as a signal telling software how statistical procedures, such as producing a chart or fitting a model, should behave. In particular, ordinal data can be represented as an `ordered.factor` in *R*, preserving a user-specified ordering in charts, tables, and models. In *Python*, `scikit-learn` supports ordinal data with the `sklearn.preprocessing.OrdinalEncoder`.
- Storage and indexing can be optimized (as in a relational database).
- The possible values a given categorical variable can take are enforced in the software (like an enum).

The third “benefit” can lead to unintended or unexpected behavior: the default behavior of data import functions in *R* (e.g., `read.csv`) is to automatically convert a text column into a `factor`. Subsequent operations on that column will assume that the only allowable values for that column are the ones originally imported, and assigning a new text value will introduce a warning and produce an `NA` (missing value). The `pandas` package in *Python* will not make such a conversion automatically. You can however specify a column as categorical explicitly in the `read_csv` function.

KEY IDEAS

- Data is typically classified in software by type.
- Data types include continuous, discrete, categorical (which includes binary), and ordinal.
- Data typing in software acts as a signal to the software on how to process the data.

Further Reading

- Data types can be confusing, since types may overlap, and the taxonomy in one software may differ from that in another. The R-Tutorial website covers the taxonomy for *R*. The *http://pandas.pydata.org/pandas-docs/stable/getting_started/basics.html#basics-dtypes* [`pandas` documentation] describes the different data types and how they can be manipulated.
- Databases are more detailed in their classification of data types, incorporating considerations of precision levels, fixed- or variable-length fields, and more; see the W3Schools guide for SQL.

Rectangular Data

The typical frame of reference for an analysis in data science is a *rectangular data* object, like a spreadsheet or database table.

KEY TERMS FOR RECTANGULAR DATA

Data frame

Rectangular data (like a spreadsheet) is the basic data structure for statistical and machine learning models.

Feature

A column in the table is commonly referred to as a *feature*.

Synonyms

attribute, input, predictor, variable

Outcome

Many data science projects involve predicting an *outcome*—often a yes/no outcome (in Table 1-1, it is "auction was competitive or not"). The *features* are sometimes used to predict the *outcome* in an experiment or study.

Synonyms

dependent variable, response, target, output

Records

A row in the table is commonly referred to as a *record*.

Synonyms

case, example, instance, observation, pattern, sample

Rectangular data is essentially a two-dimensional matrix with rows indicating records (cases) and columns indicating features (variables). The data doesn't always start in this form: unstructured data (e.g., text) must be processed and manipulated so that it can be represented as a set of features in the rectangular data (see "Elements of Structured Data"). Data in relational databases must be extracted and put into a single table for most data analysis and modeling tasks.

In Table 1-1, there is a mix of measured or counted data (e.g., duration and price), and categorical data (e.g., category and currency). As mentioned earlier, a special form of categorical variable is a binary (yes/no or 0/1) variable, seen in the rightmost column in Table 1-1—an indicator variable showing whether an auction was competitive or not.

Table 1-1. A typical data format

Category	currency	sellerRating	Duration	endDay	ClosePrice	OpenPrice	Competitive?
Music/Movie/Game	US	3249	5	Mon	0.01	0.01	0
Music/Movie/Game	US	3249	5	Mon	0.01	0.01	0
Automotive	US	3115	7	Tue	0.01	0.01	0
Automotive	US	3115	7	Tue	0.01	0.01	0
Automotive	US	3115	7	Tue	0.01	0.01	0
Automotive	US	3115	7	Tue	0.01	0.01	0
Automotive	US	3115	7	Tue	0.01	0.01	1
Automotive	US	3115	7	Tue	0.01	0.01	1

Data Frames and Indexes

Traditional database tables have one or more columns designated as an index. This can vastly improve the efficiency of certain SQL queries. In *Python*, with the `pandas` library, the basic rectangular data structure is a `DataFrame` object. By default, an automatic integer index is created for a `DataFrame` based on the order of the rows. In `pandas`, it is also possible

to set multilevel/hierarchical indexes to improve the efficiency of certain operations.

In *R*, the basic rectangular data structure is a `data.frame` object. A `data.frame` also has an implicit integer index based on the row order. While a custom key can be created through the `row.names` attribute, the native *R* `data.frame` does not support user-specified or multilevel indexes. To overcome this deficiency, two new packages are gaining widespread use: `data.table` and `dplyr`. Both support multilevel indexes and offer significant speedups in working with a `data.frame`.

> **TERMINOLOGY DIFFERENCES**
>
> Terminology for rectangular data can be confusing. Statisticians and data scientists use different terms for the same thing. For a statistician, *predictor variables* are used in a model to predict a *response* or *dependent variable*. For a data scientist, *features* are used to predict a *target*. One synonym is particularly confusing: computer scientists will use the term *sample* for a single row; a *sample* to a statistician means a collection of rows.

Nonrectangular Data Structures

There are other data structures besides rectangular data.

Time series data records successive measurements of the same variable. It is the raw material for statistical forecasting methods, and it is also a key component of the data produced by devices—the Internet of Things.

Spatial data structures, which are used in mapping and location analytics, are more complex and varied than rectangular data structures. In the *object* representation, the focus of the data is an object (e.g., a house) and

its spatial coordinates. The *field* view, by contrast, focuses on small units of space and the value of a relevant metric (pixel brightness, for example).

Graph (or network) data structures are used to represent physical, social, and abstract relationships. For example, a graph of a social network, such as Facebook or LinkedIn, may represent connections between people on the network. Distribution hubs connected by roads are an example of a physical network. Graph structures are useful for certain types of problems, such as network optimization and recommender systems.

Each of these data types has its specialized methodology in data science. The focus of this book is on rectangular data, the fundamental building block of predictive modeling.

GRAPHS IN STATISTICS

In computer science and information technology, the term *graph* typically refers to a depiction of the connections among entities, and to the underlying data structure. In statistics, *graph* is used to refer to a variety of plots and *visualizations*, not just of connections among entities, and the term applies just to the visualization, not to the data structure.

KEY IDEAS

- The basic data structure in data science is a rectangular matrix in which rows are records and columns are variables (features).
- Terminology can be confusing; there are a variety of synonyms arising from the different disciplines that contribute to data science (statistics, computer science, and information technology).

Further Reading

- Documentation on data frames in *R*
- Documentation on data frames in *Python*

Estimates of Location

Variables with measured or count data might have thousands of distinct values. A basic step in exploring your data is getting a “typical value” for each feature (variable): an estimate of where most of the data is located (i.e., its central tendency).

KEY TERMS FOR ESTIMATES OF LOCATION

Mean

The sum of all values divided by the number of values.

Synonyms

average

Weighted mean

The sum of all values times a weight divided by the sum of the weights.

Synonyms

weighted average

Median

The value such that one-half of the data lies above and below.

Synonyms

50th percentile

Percentile

The value such that *P* percent of the data lies below.

Synonyms

quantile

Weighted median

The value such that one-half of the sum of the weights lies above and below the sorted data.

Trimmed mean

The average of all values after dropping a fixed number of extreme values.

Synonyms

truncated mean

Robust

Not sensitive to extreme values.

Synonyms

resistant

Outlier

A data value that is very different from most of the data.

Synonyms

extreme value

At first glance, summarizing data might seem fairly trivial: just take the *mean* of the data (see "Mean"). In fact, while the mean is easy to compute and expedient to use, it may not always be the best measure for a central value. For this reason, statisticians have developed and promoted several alternative estimates to the mean.

METRICS AND ESTIMATES

Statisticians often use the term *estimates* for values calculated from the data at hand, to draw a distinction between what we see from the data, and the theoretical true or exact state of affairs. Data scientists and business analysts are more likely to refer to such values as a *metric*. The difference reflects the approach of statistics versus data science: accounting for uncertainty lies at the heart of the discipline of statistics, whereas concrete business or organizational objectives are the focus of data science. Hence, statisticians estimate, and data scientists measure.

Mean

The most basic estimate of location is the mean, or *average* value. The mean is the sum of all the values divided by the number of values. Consider the following set of numbers: {3 5 1 2}. The mean is (3 + 5 + 1 + 2) / 4 = 11 / 4 = 2.75. You will encounter the symbol $\bar{x}$ (pronounced "x-bar") to represent the mean of a sample from a population. The formula to compute the mean for a set of *n* values $x_1, x_2, ..., x_n$ is:

$$\text{Mean} = \bar{x} = \frac{\sum_i^n x_i}{n}$$

> **NOTE**
>
> *N* (or *n*) refers to the total number of records or observations. In statistics it is capitalized if it is referring to a population, and lowercase if it refers to a sample from a population. In data science, that distinction is not vital so you may see it both ways.

A variation of the mean is a *trimmed mean*, which you calculate by dropping a fixed number of sorted values at each end and then taking an average of the remaining values. Representing the sorted values by $x_{(1)}, x_{(2)}, ..., x_{(n)}$ where $x_{(1)}$ is the smallest value and $x_{(n)}$ the largest, the formula to compute the trimmed mean with p smallest and largest values omitted is:

$$\text{Trimmed mean} = \bar{x} = \frac{\sum_{i=p+1}^{n-p} x_{(i)}}{n - 2p}$$

A trimmed mean eliminates the influence of extreme values. For example, in international diving the top and bottom scores from five judges are dropped, and the final score is the average of the three remaining judges. This makes it difficult for a single judge to manipulate the score, perhaps to favor his country's contestant. Trimmed means are widely used, and in many cases, are preferable to use instead of the ordinary mean: see "Median and Robust Estimates" for further discussion.

Another type of mean is a *weighted mean*, which you calculate by multiplying each data value x_i by a weight w_i and dividing their sum by the sum of the weights. The formula for a weighted mean is:

$$\text{Weighted mean} = \bar{x}_w = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i}$$

There are two main motivations for using a weighted mean:

- Some values are intrinsically more variable than others, and highly variable observations are given a lower weight. For example, if we are taking the average from multiple sensors and one of the sensors is less accurate, then we might downweight the data from that sensor.
- The data collected does not equally represent the different groups that we are interested in measuring. For example, because of the way an online experiment was conducted, we may not have a set of data that accurately reflects all groups in the user base. To correct that, we can give a higher weight to the values from the groups that were underrepresented.

Median and Robust Estimates

The *median* is the middle number on a sorted list of the data. If there is an even number of data values, the middle value is one that is not actually in the data set, but rather the average of the two values that divide the sorted data into upper and lower halves. Compared to the mean, which uses all observations, the median depends only on the values in the center of the sorted data. While this might seem to be a disadvantage, since the mean is much more sensitive to the data, there are many instances in which the median is a better metric for location. Let’s say we want to look at typical household incomes in neighborhoods around Lake Washington in Seattle. In comparing the Medina neighborhood to the Windermere neighborhood, using the mean would produce very different results because Bill Gates lives in Medina. If we

use the median, it won’t matter how rich Bill Gates is—the position of the middle observation will remain the same.

For the same reasons that one uses a weighted mean, it is also possible to compute a *weighted median*. As with the median, we first sort the data, although each data value has an associated weight. Instead of the middle number, the weighted median is a value such that the sum of the weights is equal for the lower and upper halves of the sorted list. Like the median, the weighted median is robust to outliers.

OUTLIERS

The median is referred to as a *robust* estimate of location since it is not influenced by *outliers* (extreme cases) that could skew the results. An outlier is any value that is very distant from the other values in a data set. The exact definition of an outlier is somewhat subjective, although certain conventions are used in various data summaries and plots (see “Percentiles and Boxplots”). Being an outlier in itself does not make a data value invalid or erroneous (as in the previous example with Bill Gates). Still, outliers are often the result of data errors such as mixing data of different units (kilometers versus meters) or bad readings from a sensor. When outliers are the result of bad data, the mean will result in a poor estimate of location, while the median will be still be valid. In any case, outliers should be identified and are usually worthy of further investigation.

ANOMALY DETECTION

In contrast to typical data analysis, where outliers are sometimes informative and sometimes a nuisance, in *anomaly detection* the points of interest are the outliers, and the greater mass of data serves primarily to define the "normal" against which anomalies are measured.

The median is not the only robust estimate of location. In fact, a trimmed mean is widely used to avoid the influence of outliers. For example, trimming the bottom and top 10% (a common choice) of the data will provide protection against outliers in all but the smallest data sets. The trimmed mean can be thought of as a compromise between the median and the mean: it is robust to extreme values in the data, but uses more data to calculate the estimate for location.

OTHER ROBUST METRICS FOR LOCATION

Statisticians have developed a plethora of other estimators for location, primarily with the goal of developing an estimator more robust than the mean and also more *efficient* (i.e., better able to discern small location differences between data sets). While these methods are potentially useful for small data sets, they are not likely to provide added benefit for large or even moderately sized data sets.

Example: Location Estimates of Population and Murder Rates

Table 1-2 shows the first few rows in the data set containing population and murder rates (in units of murders per 100,000 people per year) for each state.

Table 1-2. A few rows of the data.frame state of population and murder rate by state

	State	Population	Murder Rate	Abbreviation
1	Alabama	4,779,736	5.7	AL
2	Alaska	710,231	5.6	AK
3	Arizona	6,392,017	4.7	AZ
4	Arkansas	2,915,918	5.6	AR
5	California	37,253,956	4.4	CA
6	Colorado	5,029,196	2.8	CO
7	Connecticut	3,574,097	2.4	CT
8	Delaware	897,934	5.8	DE

Compute the mean, trimmed mean, and median for the population using *R*:

```
> state <- read.csv('state.csv')
> mean(state[['Population']])
[1] 6162876
> mean(state[['Population']], trim=0.1)
[1] 4783697
> median(state[['Population']])
[1] 4436370
```

For mean and median we can use the `pandas` methods of the data frame. The trimmed mean requires the `trim_mean` function in `scipy.stats`.

```
state = pd.read_csv('state.csv')
state['Population'].mean()
trim_mean(state['Population'], 0.1)
state['Population'].median()
```

The mean is bigger than the trimmed mean, which is bigger than the median.

This is because the trimmed mean excludes the largest and smallest five states (`trim=0.1` drops 10% from each end). If we want to compute the average murder rate for the country, we need to use a weighted mean or median to account for different populations in the states. Since base *R* doesn't have a function for weighted median, we need to install a package such as `matrixStats`:

```
> weighted.mean(state[['Murder.Rate']], w=state[['Population']])
[1] 4.445834
> library('matrixStats')
> weightedMedian(state[['Murder.Rate']], w=state[['Population']])
[1] 4.4
```

Weighted mean is available with `NumPy`. For weighted median, we can use the specialized package wquantiles.

```
np.average(state['Murder.Rate'], weights=state['Population'])
wquantiles.median(state['Murder.Rate'], weights=state['Population'])
```

In this case, the weighted mean and median are about the same.

KEY IDEAS

- The basic metric for location is the mean, but it can be sensitive to extreme values (outlier).
- Other metrics (median, trimmed mean) are more robust.

Further Reading

- The wikipedia article on central tendency contains an extensive discussion of various measures of location.

- John Tukey's 1977 classic *Exploratory Data Analysis* (Pearson) is still widely read Tukey-1977.

Estimates of Variability

Location is just one dimension in summarizing a feature. A second dimension, *variability*, also referred to as *dispersion*, measures whether the data values are tightly clustered or spread out. At the heart of statistics lies variability: measuring it, reducing it, distinguishing random from real variability, identifying the various sources of real variability, and making decisions in the presence of it.

KEY TERMS FOR VARIABILITY METRICS

Deviations

The difference between the observed values and the estimate of location.

Synonyms

errors, residuals

Variance

The sum of squared deviations from the mean divided by $n - 1$ where n is the number of data values.

Synonyms

mean-squared-error

Standard deviation

The square root of the variance.

Synonyms

l2-norm, Euclidean norm

Mean absolute deviation

The mean of the absolute values of the deviations from the mean.

Synonyms

l1-norm, Manhattan norm

Median absolute deviation from the median

The median of the absolute values of the deviations from the median.

Range

The difference between the largest and the smallest value in a data set.

Order statistics

Metrics based on the data values sorted from smallest to biggest.

Synonyms

ranks

Percentile

The value such that P percent of the values take on this value or less and (100–P) percent take on this value or more.

Synonyms

quantile

Interquartile range

The difference between the 75th percentile and the 25th percentile.

Synonyms

IQR

Just as there are different ways to measure location (mean, median, etc.) there are also different ways to measure variability.

Standard Deviation and Related Estimates

The most widely used estimates of variation are based on the differences, or *deviations*, between the estimate of location and the observed data. For a set of data {1, 4, 4}, the mean is 3 and the median is 4. The deviations from the mean are the differences: 1 – 3 = –2, 4 – 3 = 1 , 4 – 3 = 1. These deviations tell us how dispersed the data is around the central value.

One way to measure variability is to estimate a typical value for these deviations. Averaging the deviations themselves would not tell us much —the negative deviations offset the positive ones. In fact, the sum of the deviations from the mean is precisely zero. Instead, a simple approach is to take the average of the absolute values of the deviations from the mean. In the preceding example, the absolute value of the deviations is {2 1 1} and their average is (2 + 1 + 1) / 3 = 1.33. This is known as the *mean absolute deviation* and is computed with the formula:

$$\text{Mean absolute deviation} = \frac{\sum_{i=1}^{n} |x_i - \bar{x}|}{n}$$

where $\bar{x}$ is the sample mean.

The best-known estimates for variability are the *variance* and the *standard deviation*, which are based on squared deviations. The variance is an average of the squared deviations, and the standard deviation is the square root of the variance.

$$\begin{aligned} \text{Variance} &= s^2 = \frac{\sum (x - \bar{x})^2}{n - 1} \\ \text{Standard deviation} &= s = \sqrt{\text{Variance}} \end{aligned}$$

The standard deviation is much easier to interpret than the variance since it is on the same scale as the original data. Still, with its more complicated and less intuitive formula, it might seem peculiar that the standard deviation is preferred in statistics over the mean absolute deviation. It owes its preeminence to statistical theory: mathematically, working with squared values is much more convenient than absolute values, especially for statistical models.

DEGREES OF FREEDOM, AND *N* OR *N* – 1?

In statistics books, there is always some discussion of why we have *n* – 1 in the denominator in the variance formula, instead of *n*, leading into the concept of *degrees of freedom*. This distinction is not important since *n* is generally large enough that it won't make much difference whether you divide by *n* or *n* – 1. But in case you are interested, here is the story. It is based on the premise that you want to make estimates about a population, based on a sample.

If you use the intuitive denominator of *n* in the variance formula, you will underestimate the true value of the variance and the standard deviation in the population. This is referred to as a *biased* estimate. However, if you divide by *n* – 1 instead of *n*, the variance becomes an *unbiased* estimate.

To fully explain why using *n* leads to a biased estimate involves the notion of degrees of freedom, which takes into account the number of constraints in computing an estimate. In this case, there are *n* – 1 degrees of freedom since there is one constraint: the standard deviation depends on calculating the sample mean. For most problems, data scientists do not need to worry about degrees of freedom.

Neither the variance, the standard deviation, nor the mean absolute deviation is robust to outliers and extreme values (see "Median and

Robust Estimates" for a discussion of robust estimates for location). The variance and standard deviation are especially sensitive to outliers since they are based on the squared deviations.

A robust estimate of variability is the *median absolute deviation from the median* or MAD:

$$\text{Median absolute deviation} = \text{Median}(|x_1 - m|, |x_2 - m|, ..., |x_N - m|)$$

where *m* is the median. Like the median, the MAD is not influenced by extreme values. It is also possible to compute a trimmed standard deviation analogous to the trimmed mean (see "Mean").

> **NOTE**
>
> The variance, the standard deviation, mean absolute deviation, and median absolute deviation from the median are not equivalent estimates, even in the case where the data comes from a normal distribution. In fact, the standard deviation is always greater than the mean absolute deviation, which itself is greater than the median absolute deviation. Sometimes, the median absolute deviation is multiplied by a constant scaling factor (it happens to work out to 1.4826) to put MAD on the same scale as the standard deviation in the case of a normal distribution.

Estimates Based on Percentiles

A different approach to estimating dispersion is based on looking at the spread of the sorted data. Statistics based on sorted (ranked) data are referred to as *order statistics*. The most basic measure is the *range*: the difference between the largest and smallest number. The minimum and maximum values themselves are useful to know, and helpful in identifying outliers, but the range is extremely sensitive to outliers and not very useful as a general measure of dispersion in the data.

To avoid the sensitivity to outliers, we can look at the range of the data after dropping values from each end. Formally, these types of estimates are based on differences between *percentiles*. In a data set, the *P*th percentile is a value such that at least *P* percent of the values take on this value or less and at least (100 – *P*) percent of the values take on this value or more. For example, to find the 80th percentile, sort the data. Then, starting with the smallest value, proceed 80 percent of the way to the largest value. Note that the median is the same thing as the 50th percentile. The percentile is essentially the same as a *quantile*, with quantiles indexed by fractions (so the .8 quantile is the same as the 80th percentile).

A common measurement of variability is the difference between the 25th percentile and the 75th percentile, called the *interquartile range* (or IQR). Here is a simple example: 3,1,5,3,6,7,2,9. We sort these to get 1,2,3,3,5,6,7,9. The 25th percentile is at 2.5, and the 75th percentile is at 6.5, so the interquartile range is 6.5 – 2.5 = 4. Software can have slightly differing approaches that yield different answers (see the following tip); typically, these differences are smaller.

For very large data sets, calculating exact percentiles can be computationally very expensive since it requires sorting all the data values. Machine learning and statistical software use special algorithms, such as [Zhang-Wang-2007], to get an approximate percentile that can be calculated very quickly and is guaranteed to have a certain accuracy.

PERCENTILE: PRECISE DEFINITION

If we have an even number of data (*n* is even), then the percentile is ambiguous under the preceding definition. In fact, we could take on any value between the order statistics $x_{(j)}$ and $x_{(j+1)}$ where *j* satisfies:

$$100 * \frac{j}{n} \leq P < 100 * \frac{j+1}{n}$$

Formally, the percentile is the weighted average:

$$\text{Percentile}\,(P) = (1-w)x_{(j)} + wx_{(j+1)}$$

for some weight *w* between 0 and 1. Statistical software has slightly differing approaches to choosing *w*. In fact, the *R* function `quantile` offers nine different alternatives to compute the quantile. Except for small data sets, you don't usually need to worry about the precise way a percentile is calculated. At the time of this writing, *Python*'s `numpy.quantile` only supports linear interpolation.

Example: Variability Estimates of State Population

Table 1-3 (repeated from Table 1-2, earlier, for convenience) shows the first few rows in the data set containing population and murder rates for each state.

Table 1-3. A few rows of the data.frame state of population and murder rate by state

	State	Population	Murder Rate	Abbreviation
1	Alabama	4,779,736	5.7	AL
2	Alaska	710,231	5.6	AK
3	Arizona	6,392,017	4.7	AZ
4	Arkansas	2,915,918	5.6	AR
5	California	37,253,956	4.4	CA
6	Colorado	5,029,196	2.8	CO
7	Connecticut	3,574,097	2.4	CT
8	Delaware	897,934	5.8	DE

Using *R*'s built-in functions for the standard deviation, interquartile range (IQR), and the median absolute deviation from the median (MAD), we can compute estimates of variability for the state population data:

```
> sd(state[['Population']])
[1] 6848235
> IQR(state[['Population']])
[1] 4847308
> mad(state[['Population']])
[1] 3849870
```

The `pandas` data frame provides methods for calculating standard deviation and quantiles. Using the quantiles, we can easily determine the IQR. For the robust MAD, we use the function `robust.scale.mad` from the `statsmodels` package.

```
state['Population'].std()
state['Population'].quantile(0.75) - state['Population'].quantile(0.25)
robust.scale.mad(state['Population'])
```

The standard deviation is almost twice as large as the MAD (in *R*, by default, the scale of the MAD is adjusted to be on the same scale as the mean). This is not surprising since the standard deviation is sensitive to outliers.

KEY IDEAS

- The variance and standard deviation are the most widespread and routinely reported statistics of variability.
- Both are sensitive to outliers.
- More robust metrics include mean and median absolute deviations from the mean and percentiles (quantiles).

Further Reading

- David Lane's online statistics resource has a section on percentiles.
- Kevin Davenport has a useful post on deviations from the median, and their robust properties in *R*-Bloggers.

Exploring the Data Distribution

Each of the estimates we've covered sums up the data in a single number to describe the location or variability of the data. It is also useful to explore how the data is distributed overall.

KEY TERMS FOR EXPLORING THE DISTRIBUTION

Boxplot

A plot introduced by Tukey as a quick way to visualize the distribution of data.

Synonyms

Box and whiskers plot

Frequency table

A tally of the count of numeric data values that fall into a set of intervals (bins).

Histogram

A plot of the frequency table with the bins on the x-axis and the count (or proportion) on the y-axis. While visually similar, bar charts should not be confused with histograms. See "Exploring Binary and Categorical Data" for a discussion of the difference.

Density plot

A smoothed version of the histogram, often based on a *kernel density estimate*.

Percentiles and Boxplots

In "Estimates Based on Percentiles", we explored how percentiles can be used to measure the spread of the data. Percentiles are also valuable to summarize the entire distribution. It is common to report the quartiles (25th, 50th, and 75th percentiles) and the deciles (the 10th, 20th, …, 90th percentiles). Percentiles are especially valuable to summarize the *tails* (the outer range) of the distribution. Popular culture has coined the term *one-percenters* to refer to the people in the top 99th percentile of wealth.

Table 1-4 displays some percentiles of the murder rate by state. In *R*, this would be produced by the `quantile` function:

```
quantile(state[['Murder.Rate']], p=c(.05, .25, .5, .75, .95))
   5%   25%   50%   75%   95%
1.600 2.425 4.000 5.550 6.510
```

The `pandas` data frame method `quantile` provides it in *Python*.

```
state['Murder.Rate'].quantile([0.05, 0.25, 0.5, 0.75, 0.95])
```

Table 1-4. Percentiles of murder rate by state

5%	25%	50%	75%	95%
1.60	2.42	4.00	5.55	6.51

The median is 4 murders per 100,000 people, although there is quite a bit of variability: the 5th percentile is only 1.6 and the 95th percentile is 6.51.

Boxplots, introduced by Tukey [Tukey-1977], are based on percentiles and give a quick way to visualize the distribution of data. Figure 1-2 shows a boxplot of the population by state produced by *R*:

```
boxplot(state[['Population']]/1000000, ylab='Population (millions)')
```

`Pandas` provides a number of basic exploratory plots for data frame; one of them are boxplots.

```
ax = (state['Population']/1_000_000).plot.box()
ax.set_ylabel('Population (millions)')
```

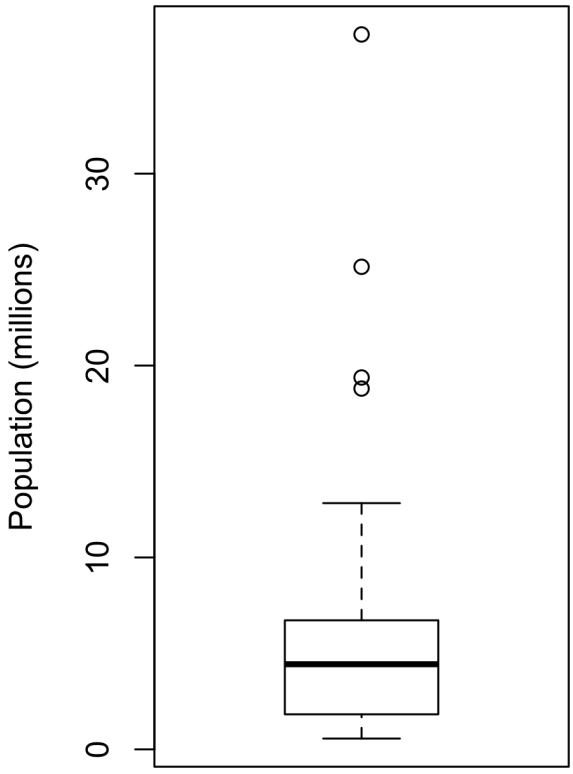

Figure 1-2. Boxplot of state populations

The top and bottom of the box are the 75th and 25th percentiles, respectively. The median is shown by the horizontal line in the box. The dashed lines, referred to as *whiskers*, extend from the top and bottom to indicate the range for the bulk of the data. There are many variations of a boxplot; see, for example, the documentation for the *R* function `boxplot` [R-base-2015]. By default, the *R* function extends the whiskers to the furthest point beyond the box, except that it will not go beyond 1.5 times the IQR. *Matplotlib* uses the same implementation; other software may use a different rule. Any data outside of the whiskers is plotted as single points.

Frequency Table and Histograms

A frequency table of a variable divides up the variable range into equally spaced segments, and tells us how many values fall in each segment.

Table 1-5 shows a frequency table of the population by state computed in *R*:

```
breaks <- seq(from=min(state[['Population']]),
              to=max(state[['Population']]), length=11)
pop_freq <- cut(state[['Population']], breaks=breaks,
              right=TRUE, include.lowest=TRUE)
table(pop_freq)
```

The function `pandas.cut` creates a series that maps the value into the segments. Using the method `value_counts` we get the frequency table.

```
binnedPopulation = pd.cut(state['Population'], 10)
binnedPopulation.value_counts()
```

Table 1-5. A frequency table of population by state

BinNumber	BinRange	Count	States
1	563,626–4,232,658	24	WY,VT,ND,AK,SD,DE,MT,RI,NH,ME,HI,ID,NE,WV,NM,NV,UT,KS,AR,MS,IA,CT,OK,OR
2	4,232,659–7,901,691	14	KY,LA,SC,AL,CO,MN,WI,MD,MO,TN,AZ,IN,MA,WA
3	7,901,692–11,570,724	6	VA,NJ,NC,GA,MI,OH
4	11,570,725–15,239,757	2	PA,IL
5	15,239,758–18,908,790	1	FL
6	18,908,791–22,577,823	1	NY
7	22,577,824–26,246,856	1	TX
8	26,246,857–29,915,889	0	
9	29,915,890–33,584,922	0	
10	33,584,923–37,253,956	1	CA

The least populous state is Wyoming, with 563,626 people (2010 Census) and the most populous is California, with 37,253,956 people. This gives us a range of 37,253,956 – 563,626 = 36,690,330, which we must divide up into equal size bins—let's say 10 bins. With 10 equal size bins, each bin will have a width of 3,669,033, so the first bin will span from 563,626 to 4,232,658. By contrast, the top bin, 33,584,923 to

37,253,956, has only one state: California. The two bins immediately below California are empty, until we reach Texas. It is important to include the empty bins; the fact that there are no values in those bins is useful information. It can also be useful to experiment with different bin sizes. If they are too large, important features of the distribution can be obscured. It they are too small, the result is too granular and the ability to see bigger pictures is lost.

NOTE

Both frequency tables and percentiles summarize the data by creating bins. In general, quartiles and deciles will have the same count in each bin (equal-count bins), but the bin sizes will be different. The frequency table, by contrast, will have different counts in the bins (equal-size bins).

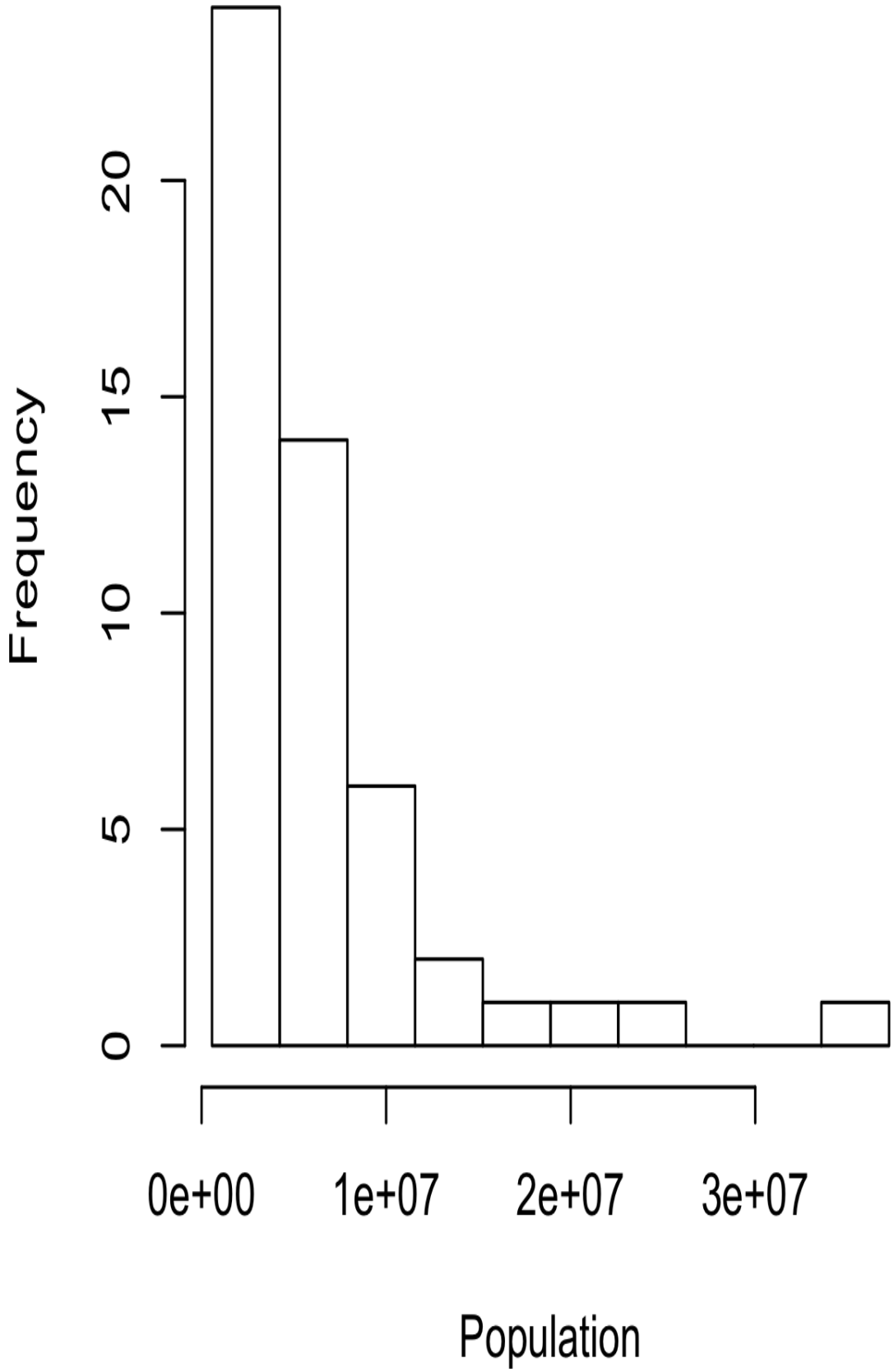

Figure 1-3. Histogram of state populations

A histogram is a way to visualize a frequency table, with bins on the x-axis and data count on the y-axis. To create a histogram corresponding to Table 1-5 in *R*, use the `hist` function with the `breaks` argument:

```
hist(state[['Population']], breaks=breaks)
```

`Pandas` supports histograms for data frames with the `DataFrame.plot.hist` method. Use the keyword argument `bins` to define the number of bins. The various plot methods return an axis object that allows further fine tuning of the visualization using `Matplotlib`.

```
ax = (state['Population'] / 1_000_000).plot.hist(figsize=(4, 4))
ax.set_xlabel('Population (millions)')
```

The histogram is shown in Figure 1-3. In general, histograms are plotted such that:

- Empty bins are included in the graph.
- Bins are equal width.
- Number of bins (or, equivalently, bin size) is up to the user.
- Bars are contiguous—no empty space shows between bars, unless there is an empty bin.

STATISTICAL MOMENTS

In statistical theory, location and variability are referred to as the first and second *moments* of a distribution. The third and fourth moments are called *skewness* and *kurtosis*. Skewness refers to whether the data is skewed to larger or smaller values and kurtosis indicates the propensity of the data to have extreme values. Generally, metrics are not used to measure skewness and kurtosis; instead, these are discovered through visual displays such as Figures 1-2 and 1-3.

Density Estimates

Related to the histogram is a density plot, which shows the distribution of data values as a continuous line. A density plot can be thought of as a smoothed histogram, although it is typically computed directly from the data through a *kernel density estimate* (see [Duong-2001] for a short tutorial). Figure 1-4 displays a density estimate superposed on a histogram. In *R*, you can compute a density estimate using the `density` function:

```
hist(state[['Murder.Rate']], freq=FALSE)
lines(density(state[['Murder.Rate']]), lwd=3, col='blue')
```

`Pandas` provides the `density` method to create a density plot. Use the argument `bw_method` to control the smoothness of the density curve.

```
ax = state['Murder.Rate'].plot.hist(density=True, xlim=[0,12],
bins=range(1,12))
state['Murder.Rate'].plot.density(ax=ax) ❶
ax.set_xlabel('Murder Rate (per 100,000)')
```

❶ plot functions often take an optional axis (`ax`) argument which will cause the plot to be added to the same graph.

A key distinction from the histogram plotted in Figure 1-3 is the scale of the y-axis: a density plot corresponds to plotting the histogram as a proportion rather than counts (you specify this in *R* using the argument `freq=FALSE`).

DENSITY ESTIMATION

Density estimation is a rich topic with a long history in statistical literature. In fact, over 20 *R* packages have been published that offer functions for density estimation. [Deng-Wickham-2011] give a comprehensive review of *R* packages, with a particular recommendation for `ASH` or `KernSmooth`. The density estimation methods in `pandas` and `scikit-learn` also offer good implementations. For many data science problems, there is no need to worry about the various types of density estimates; it suffices to use the base functions.

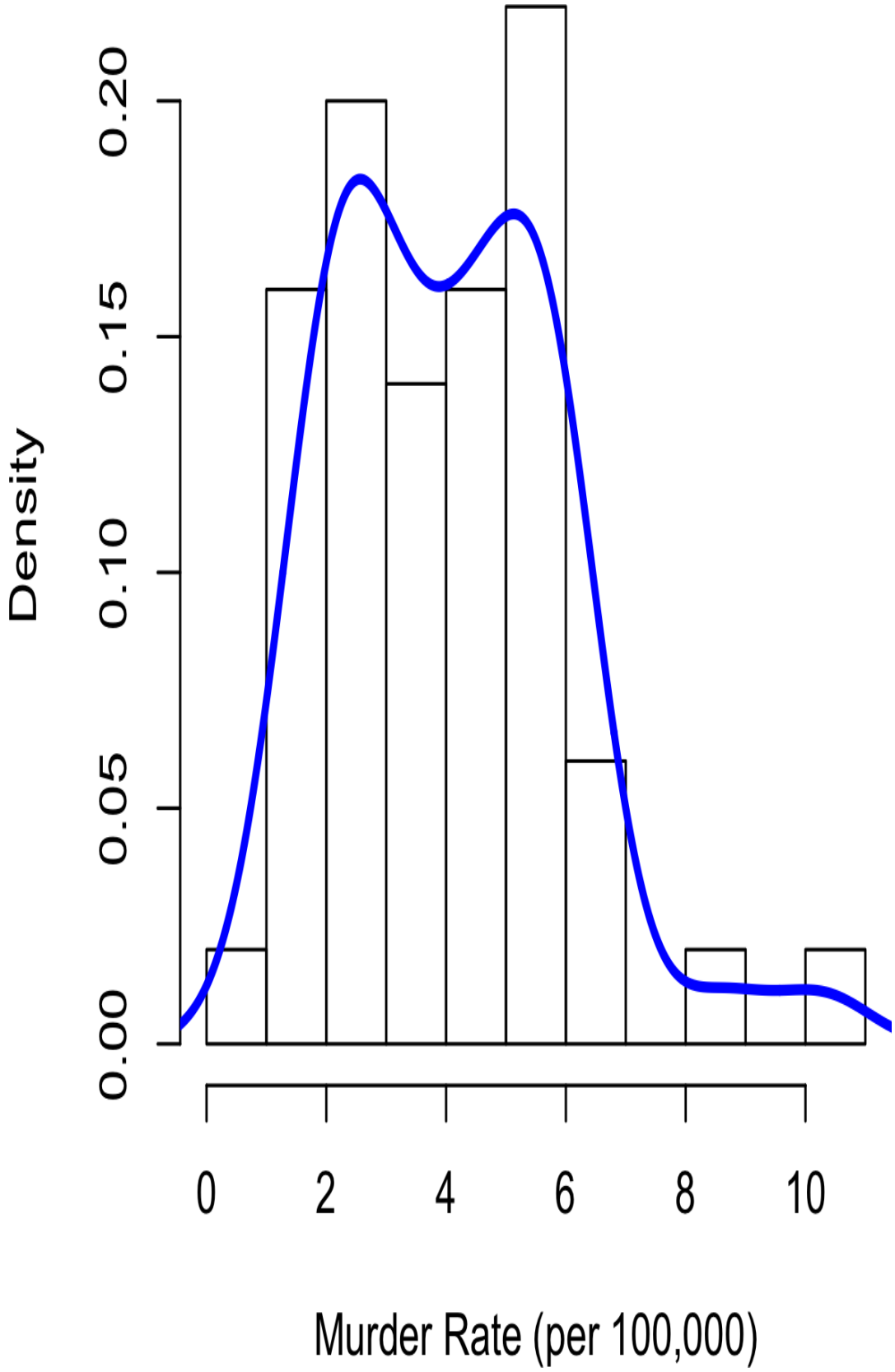

Figure 1-4. Density of state murder rates

KEY IDEAS

- A frequency histogram plots frequency counts on the y-axis and variable values on the x-axis; it gives a sense of the distribution of the data at a glance.
- A frequency table is a tabular version of the frequency counts found in a histogram.
- A boxplot—with the top and bottom of the box at the 75th and 25th percentiles, respectively—also gives a quick sense of the distribution of the data; it is often used in side-by-side displays to compare distributions.
- A density plot is a smoothed version of a histogram; it requires a function to estimate a plot based on the data (multiple estimates are possible, of course).

Further Reading

- A SUNY Oswego professor provides a step-by-step guide to creating a boxplot.
- Density estimation in *R* is covered in Henry Deng and Hadley Wickham's paper of the same name.
- *R*-Bloggers has a useful post on histograms in *R*, including customization elements, such as binning (breaks)
- *R*-Bloggers also has similar post on boxplots in *R*.
- Matthew Conlen published an interactive presentation that demonstrates the effect of choosing different kernels and bandwidth on kernel density estimates.

Exploring Binary and Categorical Data

For categorical data, simple proportions or percentages tell the story of the data.

KEY TERMS FOR EXPLORING CATEGORICAL DATA

Mode

The most commonly occurring category or value in a data set.

Expected value

When the categories can be associated with a numeric value, this gives an average value based on a category's probability of occurrence.

Bar charts

The frequency or proportion for each category plotted as bars.

Pie charts

The frequency or proportion for each category plotted as wedges in a pie.

Getting a summary of a binary variable or a categorical variable with a few categories is a fairly easy matter: we just figure out the proportion of 1s, or of the important categories. For example, Table 1-6 shows the percentage of delayed flights by the cause of delay at Dallas/Fort Worth airport since 2010. Delays are categorized as being due to factors under carrier control, air traffic control (ATC) system delays, weather, security, or a late inbound aircraft.

Table 1-6. Percentage of delays by cause at Dallas-Fort Worth airport

Carrier	ATC	Weather	Security	Inbound
23.02	30.40	4.03	0.12	42.43

Bar charts are a common visual tool for displaying a single categorical variable, often seen in the popular press. Categories are listed on the x-axis, and frequencies or proportions on the y-axis. Figure 1-5 shows the airport delays per year by cause for Dallas/Fort Worth, and it is produced with the *R* function `barplot`:

```
barplot(as.matrix(dfw) / 6, cex.axis=.5)
```

`Pandas` also supports bar charts for data frames.

```
dfw = pd.read_csv(AIRPORT_DELAYS_CSV)
100 * dfw / dfw.values.sum()
ax = dfw.transpose().plot.bar(figsize=(4, 4), legend=False)
ax.set_xlabel('Delays by cause at Dallas-Ft. Worth airport.')
```

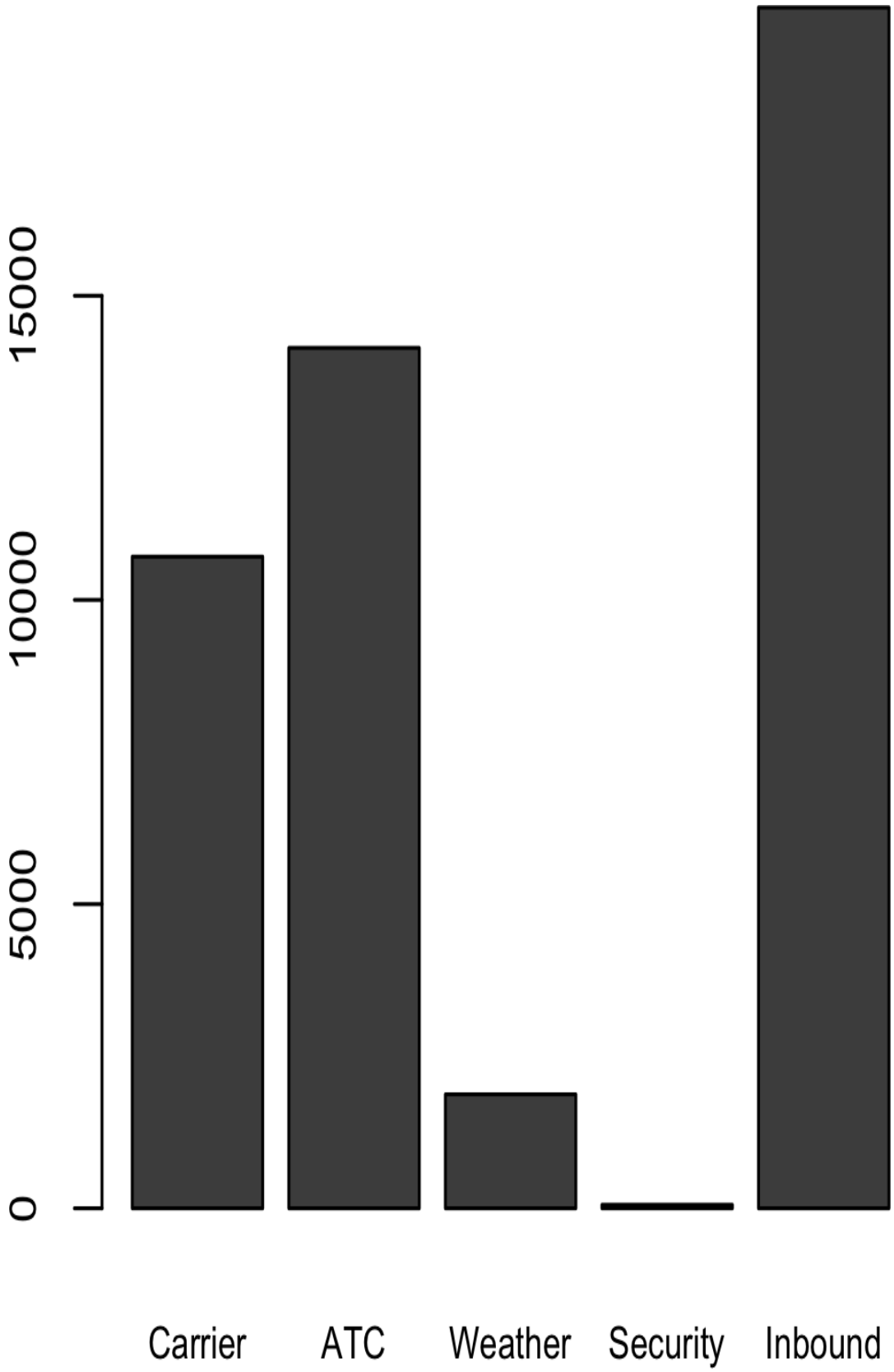

Figure 1-5. Bar plot airline delays at DFW by cause

Note that a bar chart resembles a histogram; in a bar chart the x-axis represents different categories of a factor variable, while in a histogram the x-axis represents values of a single variable on a numeric scale. In a histogram, the bars are typically shown touching each other, with gaps indicating values that did not occur in the data. In a bar chart, the bars are shown separate from one another.

Pie charts are an alternative to bar charts, although statisticians and data visualization experts generally eschew pie charts as less visually informative (see [Few-2007]).

NUMERICAL DATA AS CATEGORICAL DATA

In "Frequency Table and Histograms", we looked at frequency tables based on binning the data. This implicitly converts the numeric data to an ordered factor. In this sense, histograms and bar charts are similar, except that the categories on the x-axis in the bar chart are not ordered. Converting numeric data to categorical data is an important and widely used step in data analysis since it reduces the complexity (and size) of the data. This aids in the discovery of relationships between features, particularly at the initial stages of an analysis.

Mode

The mode is the value—or values in case of a tie—that appears most often in the data. For example, the mode of the cause of delay at Dallas/Fort Worth airport is "Inbound." As another example, in most parts of the United States, the mode for religious preference would be Christian. The mode is a simple summary statistic for categorical data, and it is generally not used for numeric data.

Expected Value

A special type of categorical data is data in which the categories represent or can be mapped to discrete values on the same scale. A marketer for a new cloud technology, for example, offers two levels of service, one priced at $300/month and another at $50/month. The marketer offers free webinars to generate leads, and the firm figures that 5% of the attendees will sign up for the $300 service, 15% for the $50 service, and 80% will not sign up for anything. This data can be summed up, for financial purposes, in a single “expected value,” which is a form of weighted mean in which the weights are probabilities.

The expected value is calculated as follows:

1. Multiply each outcome by its probability of occurring.
2. Sum these values.

In the cloud service example, the expected value of a webinar attendee is thus $22.50 per month, calculated as follows:

$$EV = (0.05)(300) + (0.15)(50) + (0.80)(0) = 22.5$$

The expected value is really a form of weighted mean: it adds the ideas of future expectations and probability weights, often based on subjective judgment. Expected value is a fundamental concept in business valuation and capital budgeting—for example, the expected value of five years of profits from a new acquisition, or the expected cost savings from new patient management software at a clinic.

Probability

We referred above to the *probability* of a value occurring. Most people have an intuitive understanding of probability, encountering the concept frequently in weather forecasts (the chance of rain) or sports analysis

(the probability of winning). Sports and games are more often expressed as odds, which are readily convertible to probabilities (if the odds that a team will win are 2 to 1, its probability of winning is 2/3). Surprisingly, though, the concept of probability can be the source of deep philosophical discussion when it comes to defining it. Fortunately, we do not need a formal mathematical or philosophical definition here. For our purposes, the probability that an event will happen is the proportion of times it will occur if the situation could be repeated over and over, countless times. Most often this is an imaginary construction, but it is an adequate operational understanding of probability.

KEY IDEAS

- Categorical data is typically summed up in proportions, and can be visualized in a bar chart.
- Categories might represent distinct things (apples and oranges, male and female), levels of a factor variable (low, medium, and high), or numeric data that has been binned.
- Expected value is the sum of values times their probability of occurrence, often used to sum up factor variable levels.

Further Reading

No statistics course is complete without a lesson on misleading graphs, which often involve bar charts and pie charts.

Correlation

Exploratory data analysis in many modeling projects (whether in data science or in research) involves examining correlation among predictors, and between predictors and a target variable. Variables X and Y (each with measured data) are said to be positively correlated if high values of X go with high values of Y, and low values of X go with low values of Y.

If high values of X go with low values of Y, and vice versa, the variables are negatively correlated.

KEY TERMS FOR CORRELATION

Correlation coefficient

A metric that measures the extent to which numeric variables are associated with one another (ranges from –1 to +1).

Correlation matrix

A table where the variables are shown on both rows and columns, and the cell values are the correlations between the variables.

Scatterplot

A plot in which the x-axis is the value of one variable, and the y-axis the value of another.

Consider these two variables, perfectly correlated in the sense that each goes from low to high:

v1: {1, 2, 3}

v2: {4, 5, 6}

The vector sum of products is 4 + 10 + 18 = 32. Now try shuffling one of them and recalculating—the vector sum of products will never be higher than 32. So this sum of products could be used as a metric; that is, the observed sum of 32 could be compared to lots of random shufflings (in fact, this idea relates to a resampling-based estimate: see [Link to Come]). Values produced by this metric, though, are not that meaningful, except by reference to the resampling distribution.

More useful is a standardized variant: the *correlation coefficient*, which gives an estimate of the correlation between two variables that always lies on the same scale. To compute *Pearson's correlation coefficient*, we

multiply deviations from the mean for variable 1 times those for variable 2, and divide by the product of the standard deviations:

$$r = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{(n-1)\, s_x s_y}$$

Note that we divide by *n* – 1 instead of *n*; see "Degrees of Freedom, and *n* or *n* – 1?" for more details. The correlation coefficient always lies between +1 (perfect positive correlation) and –1 (perfect negative correlation); 0 indicates no correlation.

Variables can have an association that is not linear, in which case the correlation coefficient may not be a useful metric. The relationship between tax rates and revenue raised is an example: as tax rates increase from 0, the revenue raised also increases. However, once tax rates reach a high level and approach 100%, tax avoidance increases and tax revenue actually declines.

Table 1-7, called a *correlation matrix*, shows the correlation between the daily returns for telecommunication stocks from July 2012 through June 2015. From the table, you can see that Verizon (VZ) and ATT (T) have the highest correlation. Level Three (LVLT), which is an infrastructure company, has the lowest correlation. Note the diagonal of 1s (the correlation of a stock with itself is 1), and the redundancy of the information above and below the diagonal.

Table 1-7. Correlation between telecommunication stock returns

	T	CTL	FTR	VZ	LVLT
T	1.000	0.475	0.328	0.678	0.279
CTL	0.475	1.000	0.420	0.417	0.287
FTR	0.328	0.420	1.000	0.287	0.260
VZ	0.678	0.417	0.287	1.000	0.242
LVLT	0.279	0.287	0.260	0.242	1.000

A table of correlations like Table 1-7 is commonly plotted to visually display the relationship between multiple variables. Figure 1-6 shows the correlation between the daily returns for major exchange traded funds (ETFs). In *R*, we can easily create this using the package `corrplot`:

```
etfs <- sp500_px[row.names(sp500_px) > '2012-07-01',
                 sp500_sym[sp500_sym$sector == 'etf', 'symbol']]
library(corrplot)
corrplot(cor(etfs), method='ellipse')
```

It is possible to create the same graph in *Python*, however there is no implementation in the common packages. However, most support the visualization of correlation matrices using heatmaps. The following code demonstrates this using the `seaborn.heatmap` package. In the accompanying source code repository, we include *Python* code to generate the more comprehensive visualization.

```
etfs = sp500_px.loc[sp500_px.index > '2012-07-01',
                    sp500_sym[sp500_sym['sector'] == 'etf']['symbol']]
sns.heatmap(etfs.corr(), vmin=-1, vmax=1,
            cmap=sns.diverging_palette(20, 220, as_cmap=True))
```

The ETFs for the S&P 500 (SPY) and the Dow Jones Index (DIA) have a high correlation. Similarly, the QQQ and the XLK, composed mostly of technology companies, are positively correlated. Defensive ETFs, such as those tracking gold prices (GLD), oil prices (USO), or market volatility (VXX) tend to be weakly or negatively correlated with the other ETFs. The orientation of the ellipse indicates whether two variables are positively correlated (ellipse is pointed to the top right) or negatively correlated (ellipse is pointed to the top left). The shading and width of the ellipse indicate the strength of the association: thinner and darker ellipses correspond to stronger relationships.

Like the mean and standard deviation, the correlation coefficient is sensitive to outliers in the data. Software packages offer robust alternatives to the classical correlation coefficient. For example, the *R* package `robust` uses the function `covRob` to compute a robust estimate of correlation. The methods in the `scikit-learn` module *sklearn.covariance* implement a variety of approaches.

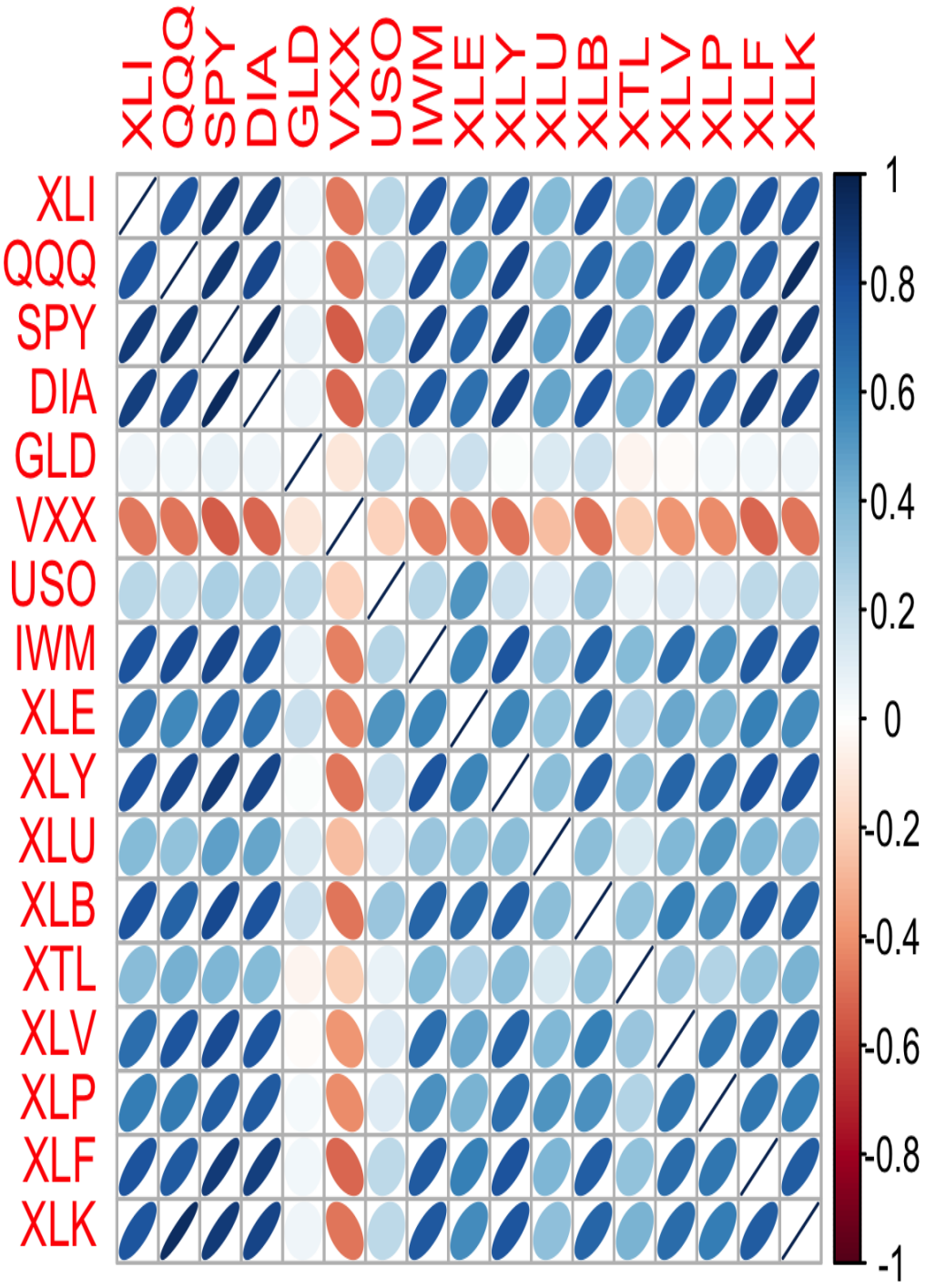

Figure 1-6. Correlation between ETF returns

OTHER CORRELATION ESTIMATES

Statisticians have long ago proposed other types of correlation coefficients, such as *Spearman's rho* or *Kendall's tau*. These are correlation coefficients based on the rank of the data. Since they work with ranks rather than values, these estimates are robust to outliers and can handle certain types of nonlinearities. However, data scientists can generally stick to Pearson's correlation coefficient, and its robust alternatives, for exploratory analysis. The appeal of rank-based estimates is mostly for smaller data sets and specific hypothesis tests.

Scatterplots

The standard way to visualize the relationship between two measured data variables is with a scatterplot. The x-axis represents one variable, the y-axis another, and each point on the graph is a record. See Figure 1-7 for a plot between the daily returns for ATT and Verizon. This is produced in *R* with the command:

```
plot(telecom$T, telecom$VZ, xlab='T', ylab='VZ')
```

The same graph can be generated in *Python* using the `pandas` scatter method.

```
ax = telecom.plot.scatter(x='T', y='VZ', figsize=(4, 4), marker='$\u25EF$')
ax.set_xlabel('ATT (T)')
ax.set_ylabel('Verizon (VZ)')
ax.axhline(0, color='grey', lw=1)
ax.axvline(0, color='grey', lw=1)
```

The returns have a strong positive relationship: on most days, both stocks go up or go down in tandem. There are very few days where one stock goes down significantly while the other stock goes up (and vice versa).

While the plot Figure 1-7 displays only 754 data points, it’s already obvious how difficult it is to identify details in the middle of the plot. We will see later how adding transparency to the points or using hexagonal binning and density plots, can help to find additional structure in the data.

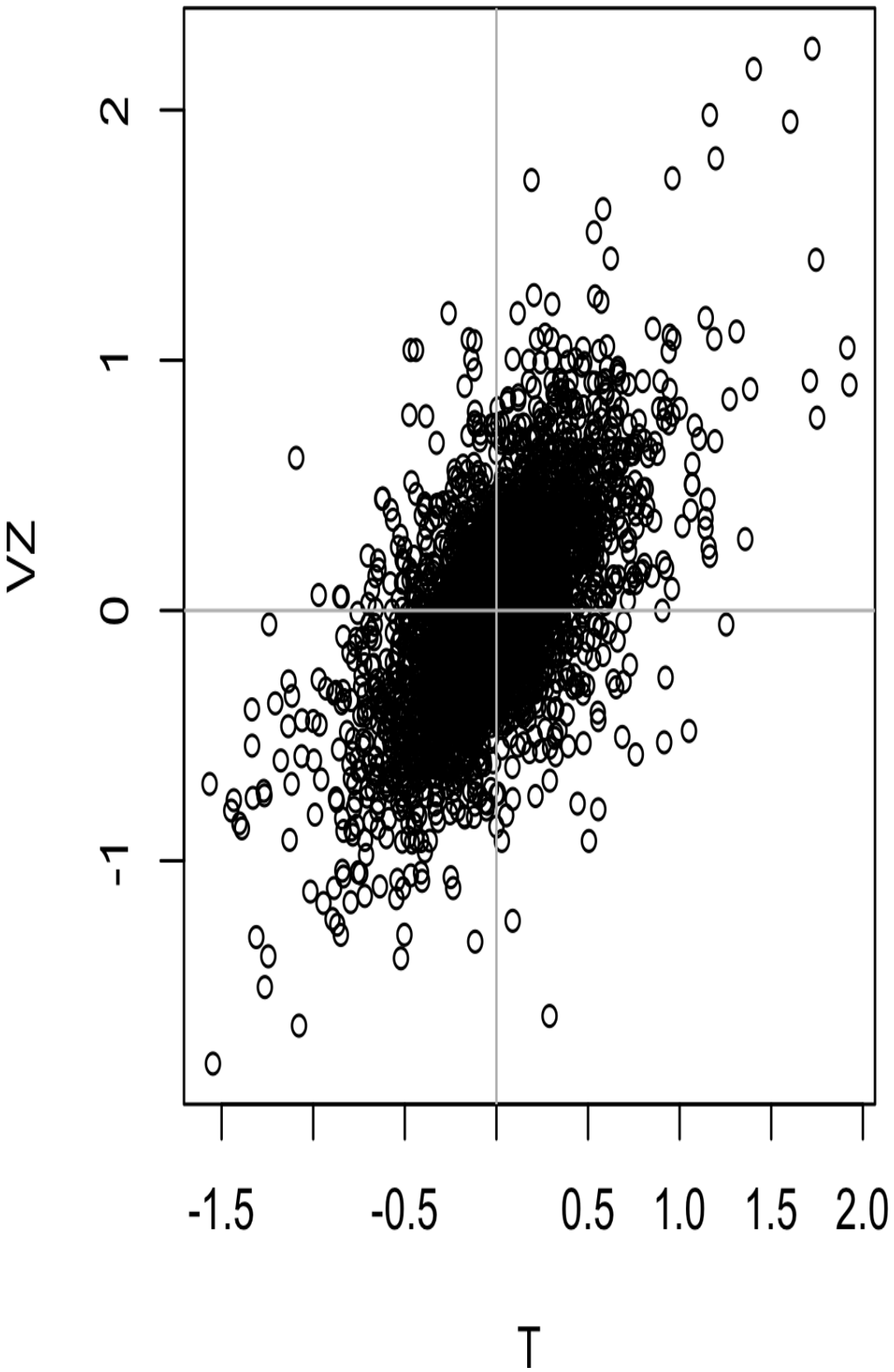

Figure 1-7. Scatterplot between returns for ATT and Verizon

KEY IDEAS FOR CORRELATION

- The correlation coefficient measures the extent to which two variables are associated with one another.
- When high values of v1 go with high values of v2, v1 and v2 are positively associated.
- When high values of v1 are associated with low values of v2, v1 and v2 are negatively associated.
- The correlation coefficient is a standardized metric so that it always ranges from –1 (perfect negative correlation) to +1 (perfect positive correlation).
- A correlation coefficient of 0 indicates no correlation, but be aware that random arrangements of data will produce both positive and negative values for the correlation coefficient just by chance.

Further Reading

Statistics, 4th ed., by David Freedman, Robert Pisani, and Roger Purves (W. W. Norton, 2007), has an excellent discussion of correlation.

Exploring Two or More Variables

Familiar estimators like mean and variance look at variables one at a time (*univariate analysis*). Correlation analysis (see "Correlation") is an important method that compares two variables (*bivariate analysis*). In this section we look at additional estimates and plots, and at more than two variables (*multivariate analysis*).

KEY TERMS FOR EXPLORING TWO OR MORE VARIABLES

Contingency tables

A tally of counts between two or more categorical variables.

Hexagonal binning

A plot of two numeric variables with the records binned into hexagons.

Contour plots

A plot showing the density of two numeric variables like a topographical map.

Violin plots

Similar to a boxplot but showing the density estimate.

Like univariate analysis, bivariate analysis involves both computing summary statistics and producing visual displays. The appropriate type of bivariate or multivariate analysis depends on the nature of the data: numeric versus categorical.

Hexagonal Binning and Contours (Plotting Numeric versus Numeric Data)

Scatterplots are fine when there is a relatively small number of data values. The plot of stock returns in Figure 1-7 involves only about 750 points. For data sets with hundreds of thousands or millions of records, a scatterplot will be too dense, so we need a different way to visualize the relationship. To illustrate, consider the data set `kc_tax`, which contains the tax-assessed values for residential properties in King County, Washington. In order to focus on the main part of the data, we strip out very expensive and very small or large residences using the `subset` function:

```
kc_tax0 <- subset(kc_tax, TaxAssessedValue < 750000 &
                  SqFtTotLiving > 100 &
```

```
                  SqFtTotLiving < 3500)
nrow(kc_tax0)
432693
```

```
kc_tax0 = kc_tax.loc[(kc_tax.TaxAssessedValue < 750000) &
                     (kc_tax.SqFtTotLiving > 100) &
                     (kc_tax.SqFtTotLiving < 3500), :]
kc_tax0.shape
(432693, 3)
```

Figure 1-8 is a *hexagon binning* plot of the relationship between the finished square feet versus the tax-assessed value for homes in King County. Rather than plotting points, which would appear as a monolithic dark cloud, we grouped the records into hexagonal bins and plotted the hexagons with a color indicating the number of records in that bin. In this chart, the positive relationship between square feet and tax-assessed value is clear. An interesting feature is the hint of a second cloud above the main cloud, indicating homes that have the same square footage as those in the main cloud, but a higher tax-assessed value.

Figure 1-8 was generated by the powerful *R* package `ggplot2`, developed by Hadley Wickham [ggplot2]. `ggplot2` is one of several new software libraries for advanced exploratory visual analysis of data; see "Visualizing Multiple Variables".

```
ggplot(kc_tax0, (aes(x=SqFtTotLiving, y=TaxAssessedValue))) +
  stat_binhex(colour='white') +
  theme_bw() +
  scale_fill_gradient(low='white', high='black') +
  labs(x='Finished Square Feet', y='Tax Assessed Value')
```

Hexagonal binning plots are readily available using the `pandas` data frame method `hexbin`.

```
ax = kc_tax0.plot.hexbin(x='SqFtTotLiving', y='TaxAssessedValue',
                         gridsize=30, sharex=False, figsize=(5, 4))
ax.set_xlabel('Finished Square Feet')
ax.set_ylabel('Tax Assessed Value')
```

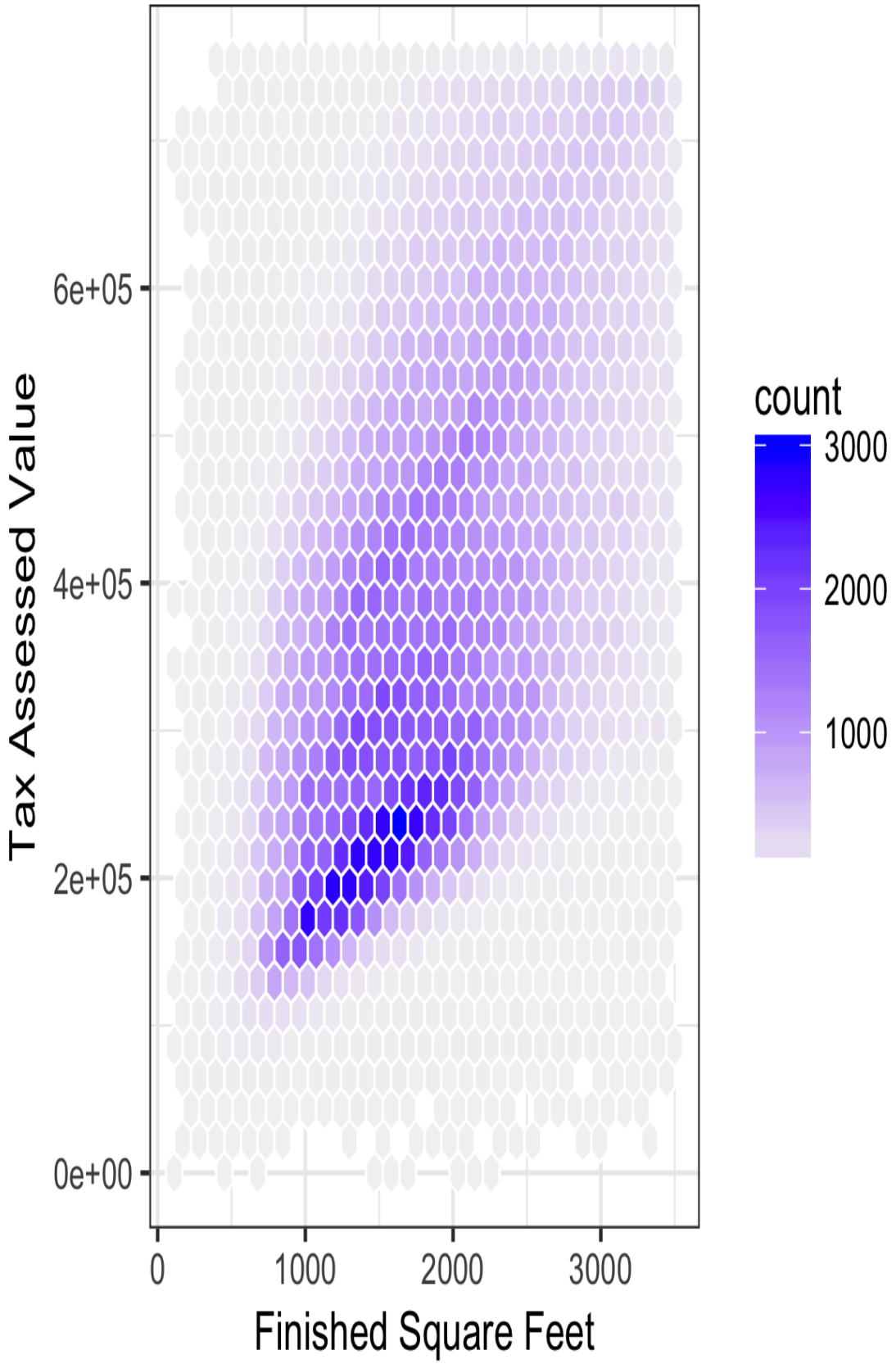

Figure 1-8. Hexagonal binning for tax-assessed value versus finished square feet

Figure 1-9 uses contours overlaid on a scatterplot to visualize the relationship between two numeric variables. The contours are essentially a topographical map to two variables; each contour band represents a specific density of points, increasing as one nears a "peak." This plot shows a similar story as Figure 1-8: there is a secondary peak "north" of the main peak. This chart was also created using `ggplot2` with the built-in `geom_density2d` function.

```
ggplot(kc_tax0, aes(SqFtTotLiving, TaxAssessedValue)) +
  theme_bw() +
  geom_point(alpha=0.1) +
  geom_density2d(colour='white') +
  labs(x='Finished Square Feet', y='Tax Assessed Value')
```

The `seaborn` `kdeplot` function creates a contour plot.

```
ax = sns.kdeplot(kc_tax0.SqFtTotLiving, kc_tax0.TaxAssessedValue, ax=ax)
ax.set_xlabel('Finished Square Feet')
ax.set_ylabel('Tax Assessed Value')
```

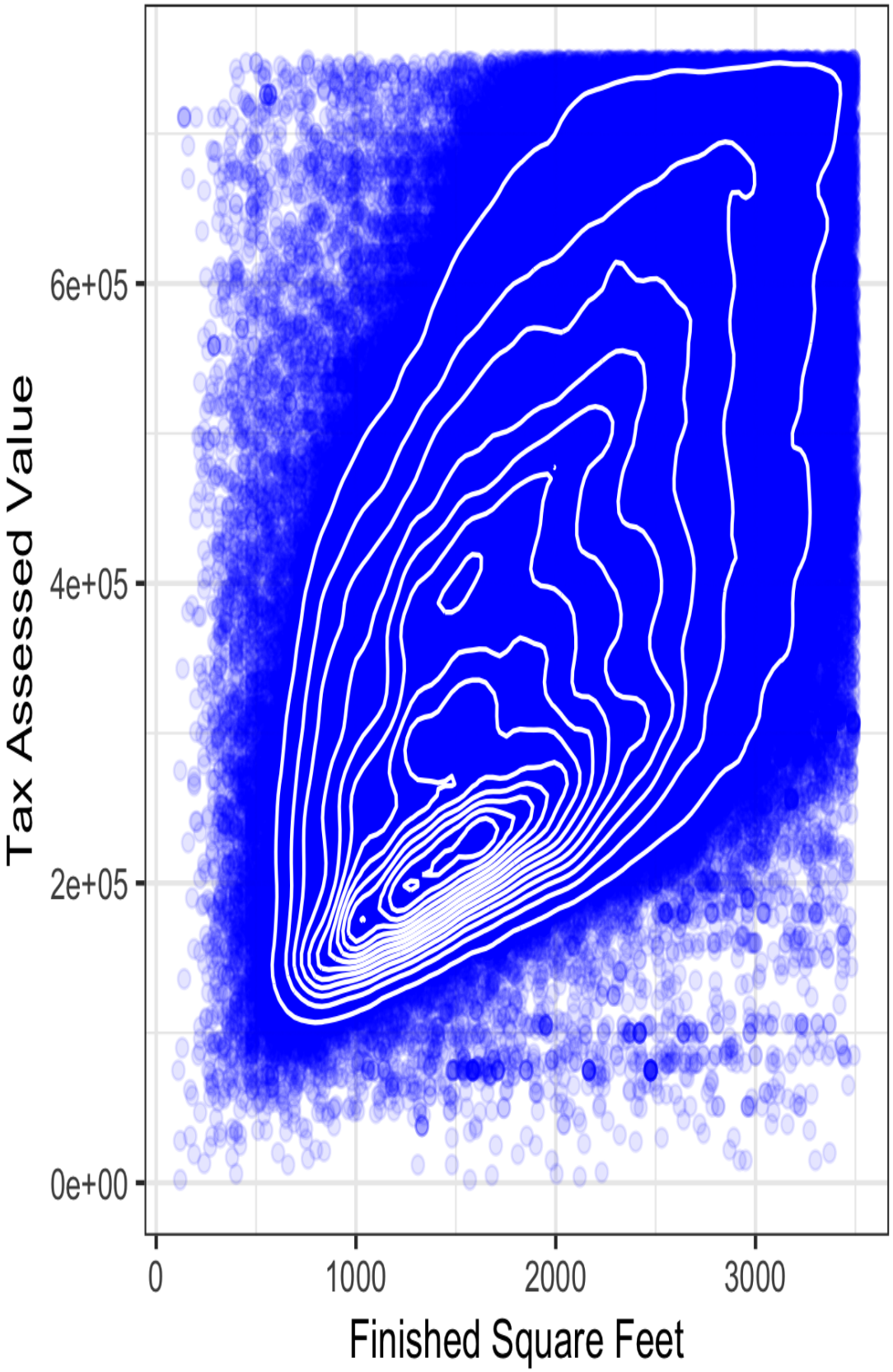

Figure 1-9. Contour plot for tax-assessed value versus finished square feet

Other types of charts are used to show the relationship between two numeric variables, including *heat maps*. Heat maps, hexagonal binning, and contour plots all give a visual representation of a two-dimensional density. In this way, they are natural analogs to histograms and density plots.

Two Categorical Variables

A useful way to summarize two categorical variables is a contingency table—a table of counts by category. Table 1-8 shows the contingency table between the grade of a personal loan and the outcome of that loan. This is taken from data provided by Lending Club, a leader in the peer-to-peer lending business. The grade goes from A (high) to G (low). The outcome is either paid off, current, late, or charged off (the balance of the loan is not expected to be collected). This table shows the count and row percentages. High-grade loans have a very low late/charge-off percentage as compared with lower-grade loans. Contingency tables can look at just counts, or also include column and total percentages. Pivot tables in Excel are perhaps the most common tool used to create contingency tables. In *R*, the `CrossTable` function in the `descr` package produces contingency tables, and the following code was used to create Table 1-8:

```
library(descr)
x_tab <- CrossTable(lc_loans$grade, lc_loans$status,
                    prop.c=FALSE, prop.chisq=FALSE, prop.t=FALSE)
```

The `pivot_table` method creates the pivot table. The `aggfunc` argument allows us to get the counts. Calculating the percentages is a bit more involved.

```
crosstab = lc_loans.pivot_table(index='grade', columns='status',
                                aggfunc=lambda x: len(x), margins=True) ❶

df = crosstab.loc['A':'G',:].copy() ❷
df.loc[:,'Charged Off':'Late'] = df.loc[:,'Charged
Off':'Late'].div(df['All'],
                                                                axis=0)
❸
df['All'] = df['All'] / sum(df['All']) ❹
perc_crosstab = df
```

❶ the `margins` keyword argument will add the column and row sums

❷ we create a copy of the pivot table ignoring the column sums

❸ divide the rows with the row sum

❹ divide the *All* column with it's sum

Table 1-8. Contingency table of loan grade and status

Grade	Charged Off	Current	Fully Paid	Late	Total
A	1562	50051	20408	469	72490
	0.022	0.690	0.282	0.006	0.161
B	5302	93852	31160	2056	132370
	0.040	0.709	0.235	0.016	0.294
C	6023	88928	23147	2777	120875
	0.050	0.736	0.191	0.023	0.268
D	5007	53281	13681	2308	74277
	0.067	0.717	0.184	0.031	0.165
E	2842	24639	5949	1374	34804
	0.082	0.708	0.171	0.039	0.077
F	1526	8444	2328	606	12904
	0.118	0.654	0.180	0.047	0.029
G	409	1990	643	199	3241
	0.126	0.614	0.198	0.061	0.007
Total	22671	321185	97316	9789	450961

Categorical and Numeric Data

Boxplots (see "Percentiles and Boxplots") are a simple way to visually compare the distributions of a numeric variable grouped according to a categorical variable. For example, we might want to compare how the percentage of flight delays varies across airlines. Figure 1-10 shows the

percentage of flights in a month that were delayed where the delay was within the carrier's control.

```
boxplot(pct_carrier_delay ~ airline, data=airline_stats, ylim=c(0, 50))
```

The `pandas` `boxplot` method takes the `by` argument that splits the dataset into groups and creates the individual boxplots.

```
ax = airline_stats.boxplot(by='airline', column='pct_carrier_delay')
ax.set_xlabel('')
ax.set_ylabel('Daily % of Delayed Flights')
plt.suptitle('')
```

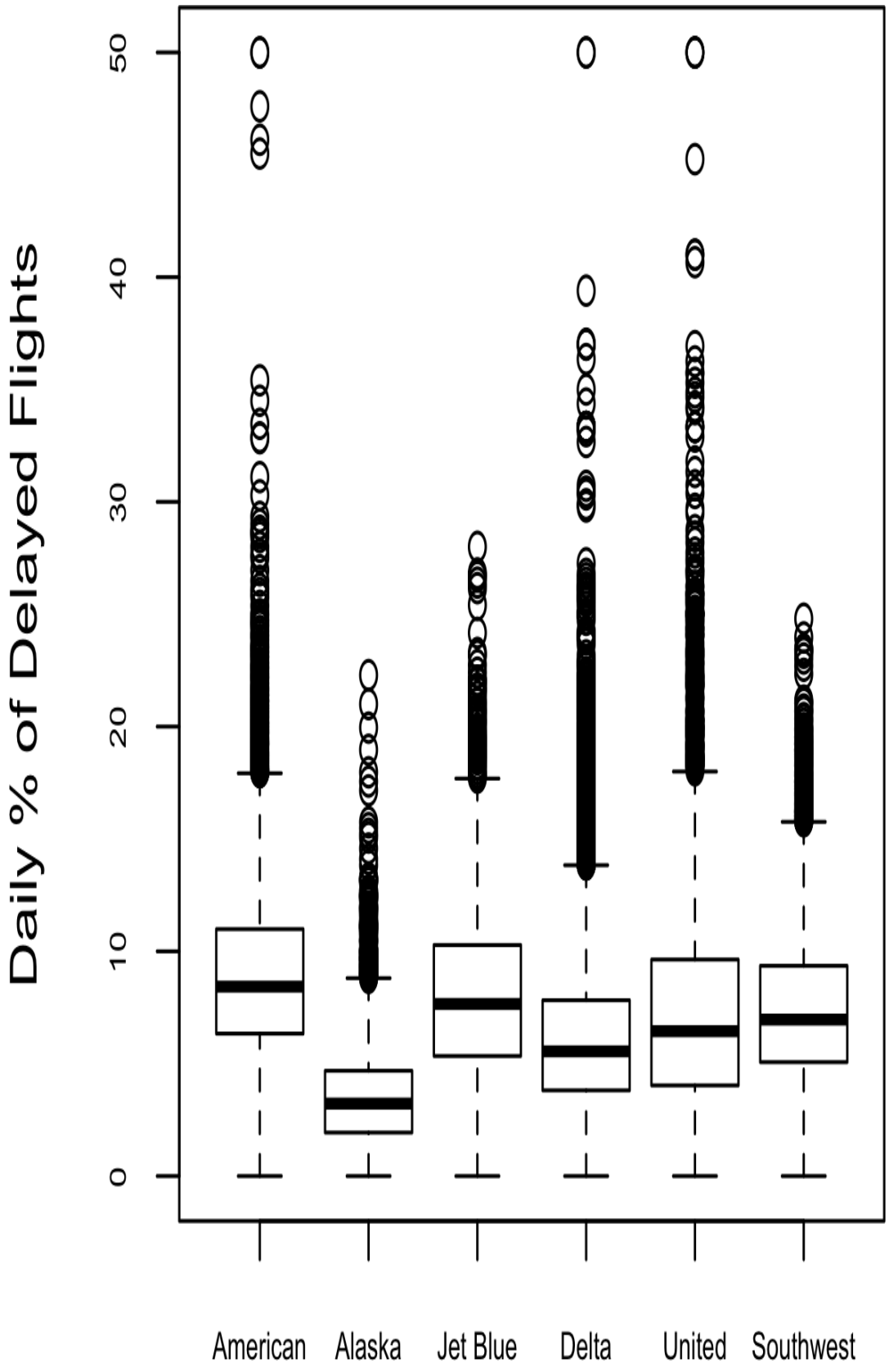

Figure 1-10. Boxplot of percent of airline delays by carrier

Alaska stands out as having the fewest delays, while American has the most delays: the lower quartile for American is higher than the upper quartile for Alaska.

A *violin plot*, introduced by [Hintze-Nelson-1998], is an enhancement to the boxplot and plots the density estimate with the density on the y-axis. The density is mirrored and flipped over and the resulting shape is filled in, creating an image resembling a violin. The advantage of a violin plot is that it can show nuances in the distribution that aren't perceptible in a boxplot. On the other hand, the boxplot more clearly shows the outliers in the data. In `ggplot2`, the function `geom_violin` can be used to create a violin plot as follows:

```
ggplot(data=airline_stats, aes(airline, pct_carrier_delay)) +
  ylim(0, 50) +
  geom_violin() +
  labs(x='', y='Daily % of Delayed Flights')
```

Violin plots are available with the `violinplot` method of the `seaborn` package.

```
ax = sns.violinplot(airline_stats.airline, airline_stats.pct_carrier_delay,
                    inner='quartile', color='white')
ax.set_xlabel('')
ax.set_ylabel('Daily % of Delayed Flights')
```

The corresponding plot is shown in Figure 1-11. The violin plot shows a concentration in the distribution near zero for Alaska, and to a lesser extent, Delta. This phenomenon is not as obvious in the boxplot. You can combine a violin plot with a boxplot by adding `geom_boxplot` to the plot (although this is best when colors are used).

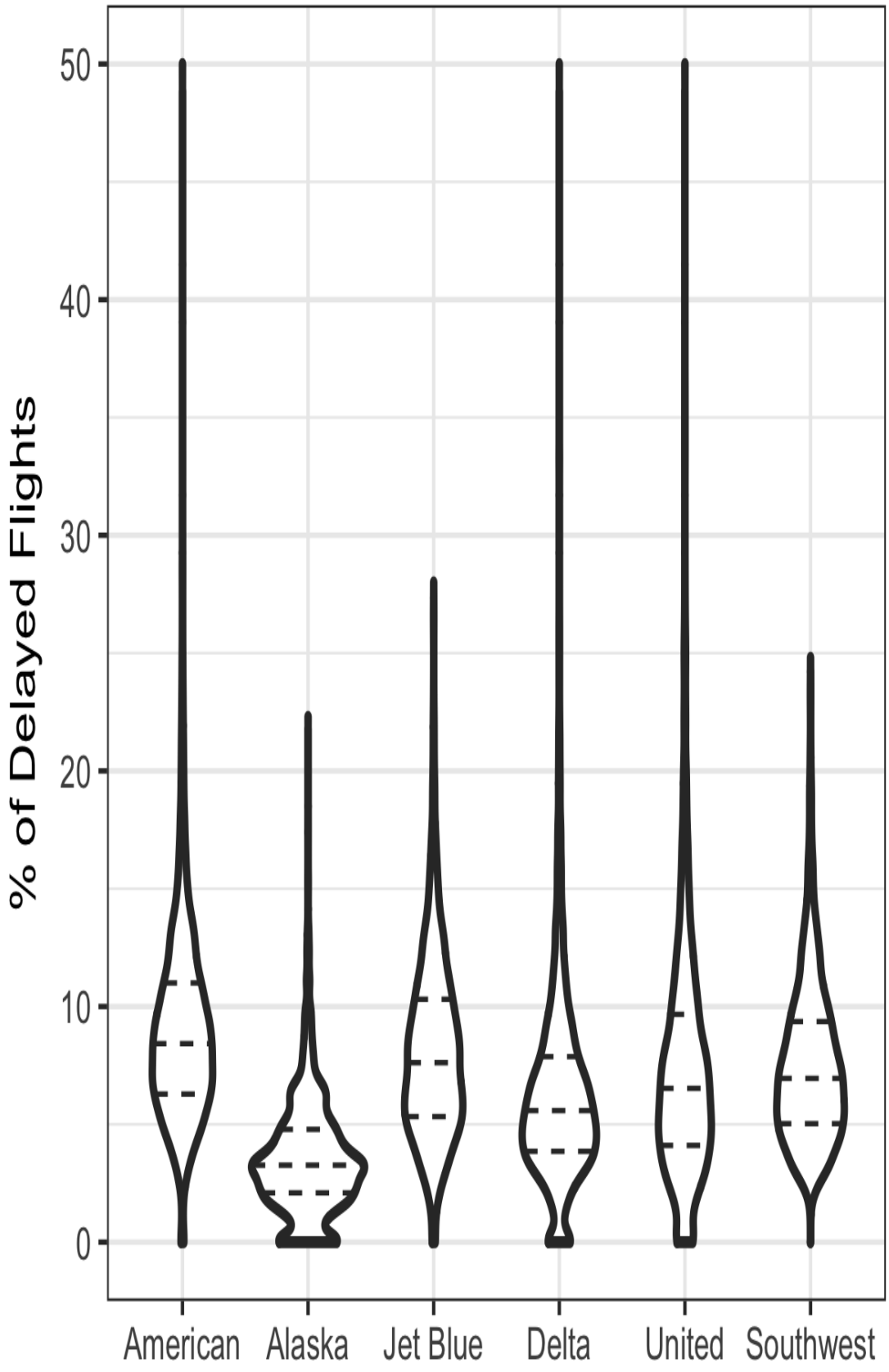

Figure 1-11. Violin plot of percent of airline delays by carrier

Visualizing Multiple Variables

The types of charts used to compare two variables—scatterplots, hexagonal binning, and boxplots—are readily extended to more variables through the notion of *conditioning*. As an example, look back at Figure 1-8, which showed the relationship between homes' finished square feet and tax-assessed values. We observed that there appears to be a cluster of homes that have higher tax-assessed value per square foot. Diving deeper, Figure 1-12 accounts for the effect of location by plotting the data for a set of zip codes. Now the picture is much clearer: tax-assessed value is much higher in some zip codes (98105, 98126) than in others (98108, 98188). This disparity gives rise to the clusters observed in Figure 1-8.

We created Figure 1-12 using `ggplot2` and the idea of *facets*, or a conditioning variable (in this case zip code):

```
ggplot(subset(kc_tax0, ZipCode %in% c(98188, 98105, 98108, 98126)),
         aes(x=SqFtTotLiving, y=TaxAssessedValue)) +
  stat_binhex(colour='white') +
  theme_bw() +
  scale_fill_gradient(low='white', high='blue') +
  labs(x='Finished Square Feet', y='Tax Assessed Value') +
  facet_wrap('ZipCode')
```

Most *Python* packages base their visualizations on `Matplotlib`. While it is in principle possible to create facetted graphs using `Matplotlib`, the code can get complicated. Fortunately, `seaborn` has a relatively straightforward way of creating these graphs.

```
zip_codes = [98188, 98105, 98108, 98126]
kc_tax_zip = kc_tax0.loc[kc_tax0.ZipCode.isin(zip_codes),:]
kc_tax_zip
```

```
def hexbin(x, y, color, **kwargs):
    cmap = sns.light_palette(color, as_cmap=True)
    plt.hexbin(x, y, gridsize=25, cmap=cmap, **kwargs)

g = sns.FacetGrid(kc_tax_zip, col='ZipCode', col_wrap=2)
g.map(hexbin, 'SqFtTotLiving', 'TaxAssessedValue', extent=[0, 3500, 0,
700000])
g.set_axis_labels('Finished Square Feet', 'Tax Assessed Value')
g.set_titles('Zip code {col_name:.0f}')
```

The `map` method calls the hexbin methods with subsets of the original dataset for the different zip codes.

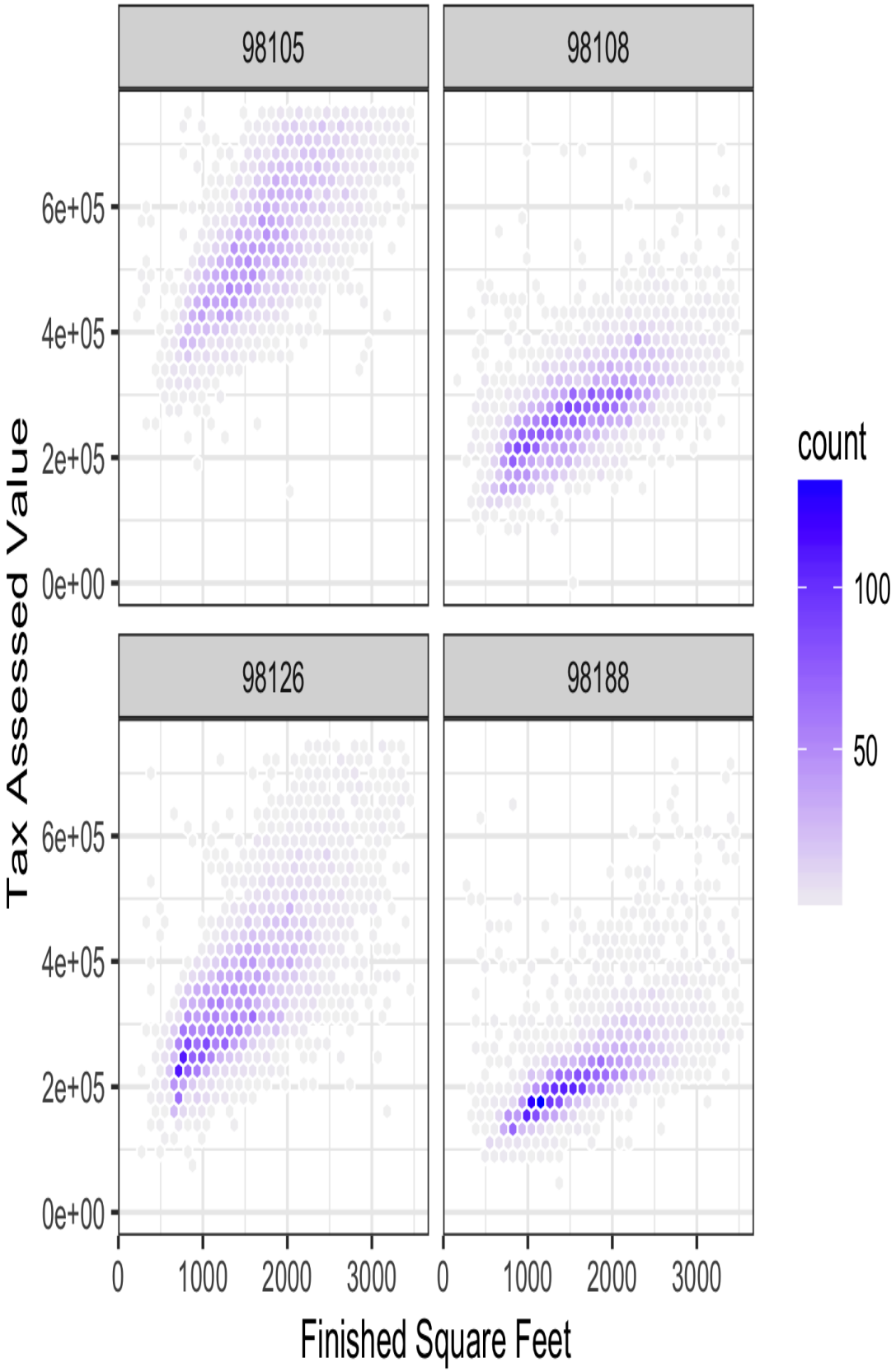

Figure 1-12. Tax-assessed value versus finished square feet by zip code

The concept of conditioning variables in a graphics system was pioneered with *Trellis graphics*, developed by Rick Becker, Bill Cleveland, and others at Bell Labs [Trellis-Graphics]. This idea has propagated to various modern graphics systems, such as the `lattice` [lattice] and `ggplot2` packages in *R* and the `seaborn` [seaborn] and `Bokeh` [bokeh] modules in *Python*. Conditioning variables are also integral to business intelligence platforms such as Tableau and Spotfire. With the advent of vast computing power, modern visualization platforms have moved well beyond the humble beginnings of exploratory data analysis. However, key concepts and tools developed over the years still form a foundation for these systems.

KEY IDEAS

- Hexagonal binning and contour plots are useful tools that permit graphical examination of two numeric variables at a time, without being overwhelmed by huge amounts of data.
- Contingency tables are the standard tool for looking at the counts of two categorical variables.
- Boxplots and violin plots allow you to plot a numeric variable against a categorical variable.

Further Reading

- *Modern Data Science with R*, by Benjamin Baumer, Daniel Kaplan, and Nicholas Horton (CRC Press, 2017), has an excellent presentation of "a grammar for graphics" (the "gg" in `ggplot`).
- *Ggplot2: Elegant Graphics for Data Analysis*, by Hadley Wickham, is an excellent resource from the creator of `ggplot2` (Springer, 2009).
- Josef Fruehwald has a web-based tutorial on `ggplot2`.

Summary

With the development of exploratory data analysis (EDA), pioneered by John Tukey, statistics set a foundation that was a precursor to the field of data science. The key idea of EDA is that the first and most important step in any project based on data is to *look at the data*. By summarizing and visualizing the data, you can gain valuable intuition and understanding of the project.

This chapter has reviewed concepts ranging from simple metrics, such as estimates of location and variability, to rich visual displays to explore the relationships between multiple variables, as in Figure 1-12. The diverse set of tools and techniques being developed by the open source community, combined with the expressiveness of the *R* and *Python* languages, has created a plethora of ways to explore and analyze data. Exploratory analysis should be a cornerstone of any data science project.

Chapter 2. Data and Sampling Distributions

A popular misconception holds that the era of big data means the end of a need for sampling. In fact, the proliferation of data of varying quality and relevance reinforces the need for sampling as a tool to work efficiently with a variety of data and to minimize bias. Even in a big data project, predictive models are typically developed and piloted with samples. Samples are also used in tests of various sorts (e.g., pricing, web treatments).

Figure 2-1 shows a schematic that underpins the concepts in this chapter. The lefthand side represents a population that, in statistics, is assumed to follow an underlying but *unknown* distribution. The only thing available is the *sample* data and its empirical distribution, shown on the righthand side. To get from the lefthand side to the righthand side, a *sampling* procedure is used (represented by an arrow). Traditional statistics focused very much on the lefthand side, using theory based on strong assumptions about the population. Modern statistics has moved to the righthand side, where such assumptions are not needed.

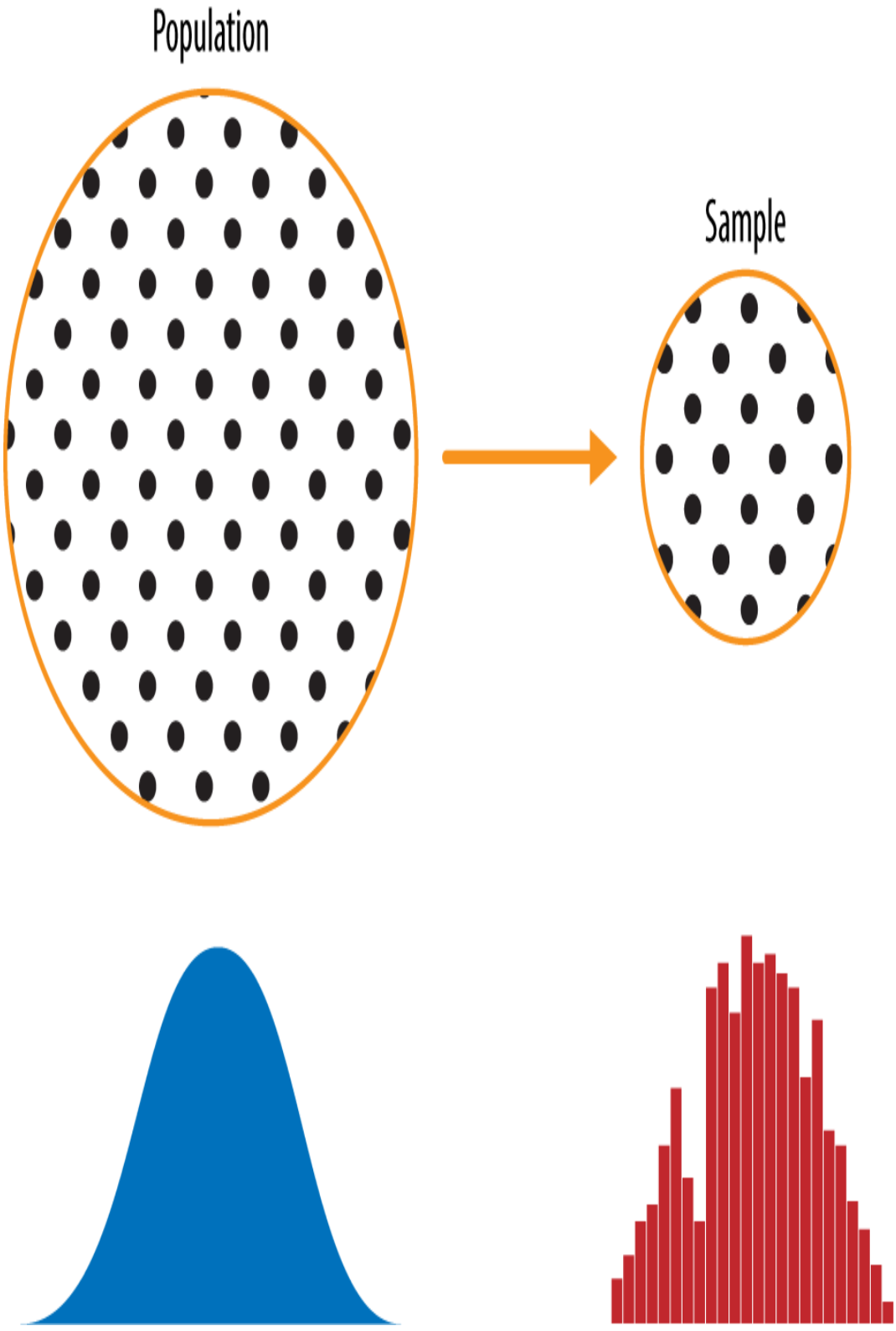

Figure 2-1. Population versus sample

In general, data scientists need not worry about the theoretical nature of the lefthand side, and instead should focus on the sampling procedures and the data at hand. There are some notable exceptions. Sometimes data is generated from a physical process that can be modeled. The simplest example is flipping a coin: this follows a binomial distribution. Any real-life binomial situation (buy or don't buy, fraud or no fraud, click or don't click) can be modeled effectively by a coin (with modified probability of landing heads, of course). In these cases, we can gain additional insight by using our understanding of the population.

Random Sampling and Sample Bias

A *sample* is a subset of data from a larger data set; statisticians call this larger data set the *population*. A population in statistics is not the same thing as in biology—it is a large, defined but sometimes theoretical or imaginary, set of data.

KEY TERMS FOR RANDOM SAMPLING

Sample

A subset from a larger data set.

Population

The larger data set or idea of a data set.

N (n)

The size of the population (sample).

Random sampling

Drawing elements into a sample at random.

Stratified sampling

Dividing the population into strata and randomly sampling from each strata.

Stratum (pl strata)

A homogeneous subgroup of a population with common characteristics.

Simple random sample

The sample that results from random sampling without stratifying the population.

Sample bias

A sample that misrepresents the population.

Random sampling is a process in which each available member of the population being sampled has an equal chance of being chosen for the sample at each draw. The sample that results is called a *simple random sample*. Sampling can be done *with replacement*, in which observations are put back in the population after each draw for possible future reselection. Or it can be done *without replacement*, in which case observations, once selected, are unavailable for future draws.

Data quality often matters more than data quantity when making an estimate or a model based on a sample. Data quality in data science involves completeness, consistency of format, cleanliness, and accuracy of individual data points. Statistics adds the notion of *representativeness*.

The classic example is the *Literary Digest* poll of 1936 that predicted a victory of Alf Landon against Franklin Roosevelt. The *Literary Digest*, a leading periodical of the day, polled its entire subscriber base, plus additional lists of individuals, a total of over 10 million, and predicted a landslide victory for Landon. George Gallup, founder of the Gallup Poll, conducted biweekly polls of just 2,000, and accurately predicted a Roosevelt victory. The difference lay in the selection of those polled.

The *Literary Digest* opted for quantity, paying little attention to the method of selection. They ended up polling those with relatively high socioeconomic status (their own subscribers, plus those who, by virtue of owning luxuries like telephones and automobiles, appeared in marketers' lists). The result was *sample bias*; that is, the sample was different in some meaningful nonrandom way from the larger population it was meant to represent. The term *nonrandom* is important—hardly any sample, including random samples, will be exactly representative of the population. Sample bias occurs when the difference is meaningful, and can be expected to continue for other samples drawn in the same way as the first.

SELF-SELECTION SAMPLING BIAS

The reviews of restaurants, hotels, cafes, and so on that you read on social media sites like Yelp are prone to bias because the people submitting them are not randomly selected; rather, they themselves have taken the initiative to write. This leads to self-selection bias—the people motivated to write reviews may be those who had poor experiences, may have an association with the establishment, or may simply be a different type of person from those who do not write reviews. Note that while self-selection samples can be unreliable indicators of the true state of affairs, they may be more reliable in simply comparing one establishment to a similar one; the same self-selection bias might apply to each.

Bias

Statistical bias refers to measurement or sampling errors that are systematic and produced by the measurement or sampling process. An important distinction should be made between errors due to random chance, and errors due to bias. Consider the physical process of a gun shooting at a target. It will not hit the absolute center of the target every time, or even much at all. An unbiased process will produce error, but it is random and does not tend strongly in any direction (see Figure 2-2). The results shown in Figure 2-3 show a biased process—there is still random error in both the x and y direction, but there is also a bias. Shots tend to fall in the upper-right quadrant.

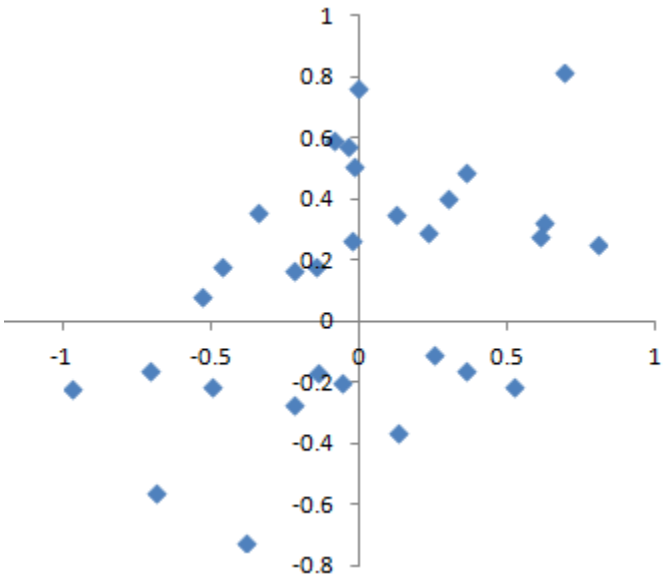

Figure 2-2. Scatterplot of shots from a gun with true aim

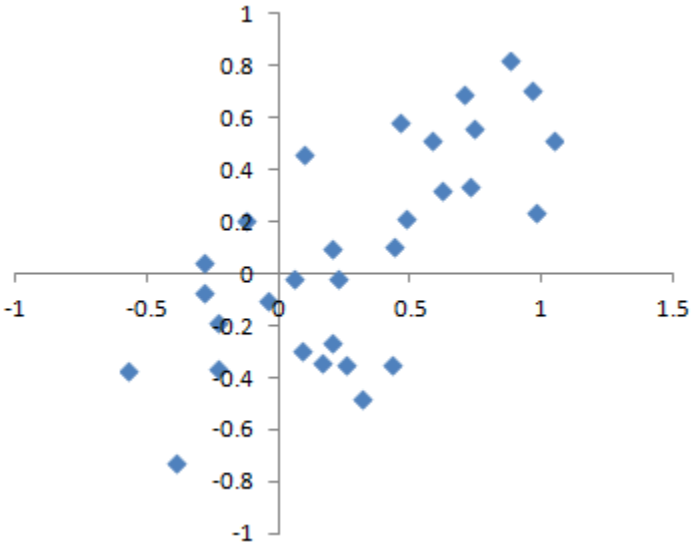

Figure 2-3. Scatterplot of shots from a gun with biased aim

Bias comes in different forms, and may be observable or invisible. When a result does suggest bias (e.g., by reference to a benchmark or actual values), it is often an indicator that a statistical or machine learning model has been misspecified, or an important variable left out.

Random Selection

To avoid the problem of sample bias that led the *Literary Digest* to predict Landon over Roosevelt, George Gallup (shown in Figure 2-4) opted for more scientifically chosen methods to achieve a sample that was representative of the US voter. There are now a variety of methods to achieve representativeness, but at the heart of all of them lies *random sampling*.

Figure 2-4. George Gallup, catapulted to fame by the Literary Digest's "big data" failure

Random sampling is not always easy. Proper definition of an accessible population is key. Suppose we want to generate a representative profile of customers and we need to conduct a pilot customer survey. The survey needs to be representative but is labor intensive.

First we need to define who a customer is. We might select all customer records where purchase amount > 0. Do we include all past customers? Do we include refunds? Internal test purchases? Resellers? Both billing agent and customer?

Next we need to specify a sampling procedure. It might be “select 100 customers at random.” Where a sampling from a flow is involved (e.g., real-time customer transactions or web visitors), timing considerations may be important (e.g., a web visitor at 10 a.m. on a weekday may be different from a web visitor at 10 p.m. on a weekend).

In *stratified sampling*, the population is divided up into *strata*, and random samples are taken from each stratum. Political pollsters might seek to learn the electoral preferences of whites, blacks, and Hispanics. A simple random sample taken from the population would yield too few blacks and Hispanics, so those strata could be overweighted in stratified sampling to yield equivalent sample sizes.

Size versus Quality: When Does Size Matter?

In the era of big data, it is sometimes surprising that smaller is better. Time and effort spent on random sampling not only reduce bias, but also allow greater attention to data exploration and data quality. For example, missing data and outliers may contain useful information. It might be prohibitively expensive to track down missing values or evaluate outliers in millions of records, but doing so in a sample of several thousand records may be feasible. Data plotting and manual inspection bog down if there is too much data.

So when *are* massive amounts of data needed?

The classic scenario for the value of big data is when the data is not only big, but sparse as well. Consider the search queries received by

Google, where columns are terms, rows are individual search queries, and cell values are either 0 or 1, depending on whether a query contains a term. The goal is to determine the best predicted search destination for a given query. There are over 150,000 words in the English language, and Google processes over 1 trillion queries per year. This yields a huge matrix, the vast majority of whose entries are "0."

This is a true big data problem—only when such enormous quantities of data are accumulated can effective search results be returned for most queries. And the more data accumulates, the better the results. For popular search terms this is not such a problem—effective data can be found fairly quickly for the handful of extremely popular topics trending at a particular time. The real value of modern search technology lies in the ability to return detailed and useful results for a huge variety of search queries, including those that occur only with a frequency, say, of one in a million.

Consider the search phrase "Ricky Ricardo and Little Red Riding Hood." In the early days of the internet, this query would probably have returned results on Ricky Ricardo the band leader, the television show *I Love Lucy* in which he starred, and the children's story *Little Red Riding Hood*. Later, now that trillions of search queries have been accumulated, this search query returns the exact *I Love Lucy* episode in which Ricky narrates, in dramatic fashion, the Little Red Riding Hood story to his infant son in a comic mix of English and Spanish.

Keep in mind that the number of actual *pertinent* records—ones in which this exact search query, or something very similar, appears (together with information on what link people ultimately clicked on)—might need only be in the thousands to be effective. However, many trillions of data points are needed in order to obtain these pertinent records (and random sampling, of course, will not help). See also "Long-Tailed Distributions".

Sample Mean versus Population Mean

The symbol $\bar{x}$ (pronounced x-bar) is used to represent the mean of a sample from a population, whereas μ is used to represent the mean of a population. Why make the distinction? Information about samples is observed, and information about large populations is often inferred from smaller samples. Statisticians like to keep the two things separate in the symbology.

KEY IDEAS

- Even in the era of big data, random sampling remains an important arrow in the data scientist's quiver.
- Bias occurs when measurements or observations are systematically in error because they are not representative of the full population.
- Data quality is often more important than data quantity, and random sampling can reduce bias and facilitate quality improvement that would be prohibitively expensive.

Further Reading

- A useful review of sampling procedures can be found in Ronald Fricker's chapter "Sampling Methods for Web and E-mail Surveys," found in the *Sage Handbook of Online Research Methods*. This chapter includes a review of the

modifications to random sampling that are often used for practical reasons of cost or feasibility.

- The story of the *Literary Digest* poll failure can be found on the Capital Century website.

Selection Bias

To paraphrase Yogi Berra, “If you don’t know what you’re looking for, look hard enough and you’ll find it.”

Selection bias refers to the practice of selectively choosing data—consciously or unconsciously—in a way that that leads to a conclusion that is misleading or ephemeral.

KEY TERMS

Bias

Systematic error.

Data snooping

Extensive hunting through data in search of something interesting.

Vast search effect

Bias or nonreproducibility resulting from repeated data modeling, or modeling data with large numbers of predictor variables.

If you specify a hypothesis and conduct a well-designed experiment to test it, you can have high confidence in the conclusion. Such is often not the case, however. Often, one looks at available data and tries to discern patterns. But is the pattern for real, or just the product of *data snooping*—that is, extensive hunting through the data until

something interesting emerges? There is a saying among statisticians: “If you torture the data long enough, sooner or later it will confess.”

The difference between a phenomenon that you verify when you test a hypothesis using an experiment, versus a phenomenon that you discover by perusing available data, can be illuminated with the following thought experiment.

Imagine that someone tells you she can flip a coin and have it land heads on the next 10 tosses. You challenge her (the equivalent of an experiment), and she proceeds to toss it 10 times, all landing heads. Clearly you ascribe some special talent to her—the probability that 10 coin tosses will land heads just by chance is 1 in 1,000.

Now imagine that the announcer at a sports stadium asks the 20,000 people in attendance each to toss a coin 10 times, and report to an usher if they get 10 heads in a row. The chance that *somebody* in the stadium will get 10 heads is extremely high (more than 99%—it’s 1 minus the probability that nobody gets 10 heads). Clearly, selecting, after the fact, the person (or persons) who gets 10 heads at the stadium does not indicate they have any special talent—it’s most likely luck.

Since repeated review of large data sets is a key value proposition in data science, selection bias is something to worry about. A form of selection bias of particular concern to data scientists is what John Elder (founder of Elder Research, a respected data mining consultancy) calls the *vast search effect*. If you repeatedly run different models and ask different questions with a large data set, you

are bound to find something interesting. Is the result you found truly something interesting, or is it the chance outlier?

We can guard against this by using a holdout set, and sometimes more than one holdout set, against which to validate performance. Elder also advocates the use of what he calls *target shuffling* (a permutation test, in essence) to test the validity of predictive associations that a data mining model suggests.

Typical forms of selection bias in statistics, in addition to the vast search effect, include nonrandom sampling (see *sampling bias*), cherry-picking data, selection of time intervals that accentuate a particular statistical effect, and stopping an experiment when the results look “interesting.”

Regression to the Mean

Regression to the mean refers to a phenomenon involving successive measurements on a given variable: extreme observations tend to be followed by more central ones. Attaching special focus and meaning to the extreme value can lead to a form of selection bias.

Sports fans are familiar with the “rookie of the year, sophomore slump” phenomenon. Among the athletes who begin their career in a given season (the rookie class), there is always one who performs better than all the rest. Generally, this “rookie of the year” does not do as well in his second year. Why not?

In nearly all major sports, at least those played with a ball or puck, there are two elements that play a role in overall performance:

- Skill
- Luck

Regression to the mean is a consequence of a particular form of selection bias. When we select the rookie with the best performance, skill and good luck are probably contributing. In his next season, the skill will still be there but, in most cases, the luck will not, so his performance will decline—it will regress. The phenomenon was first identified by Francis Galton in 1886 [Galton-1886], who wrote of it in connection with genetic tendencies; for example, the children of extremely tall men tend not to be as tall as their father (see Figure 2-5).

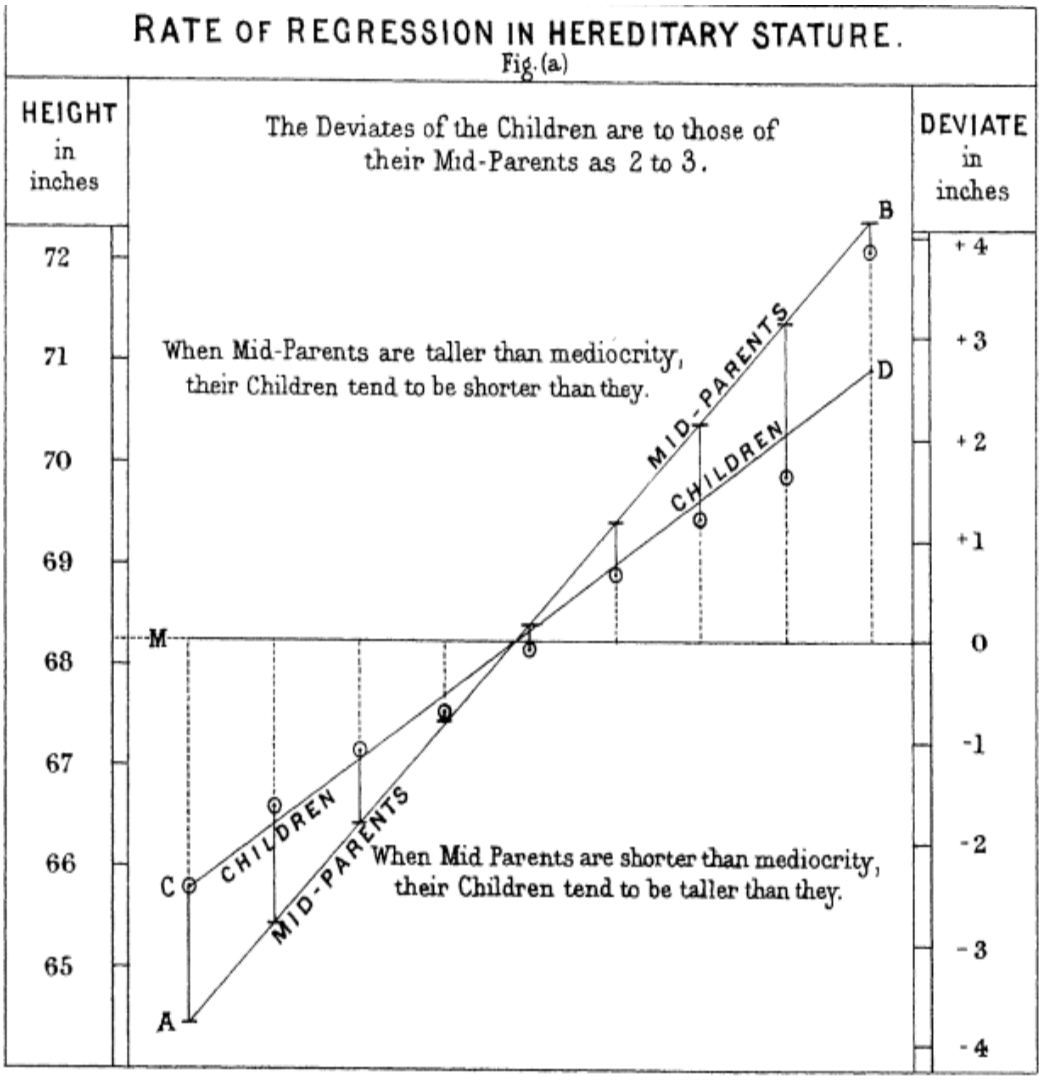

Figure 2-5. Galton's study that identified the phenomenon of regression to the mean

WARNING

Regression to the mean, meaning to "go back," is distinct from the statistical modeling method of linear regression, in which a linear relationship is estimated between predictor variables and an outcome variable.

KEY IDEAS

- Specifying a hypothesis, then collecting data following randomization and random sampling principles, ensures against bias.
- All other forms of data analysis run the risk of bias resulting from the data collection/analysis process (repeated running of models in data mining, data snooping in research, and after-the-fact selection of interesting events).

Further Reading

- Christopher J. Pannucci and Edwin G. Wilkins' article "Identifying and Avoiding Bias in Research" in (surprisingly) *Plastic and Reconstructive Surgery* (August 2010) has an excellent review of various types of bias that can enter into research, including selection bias.
- Michael Harris's article "Fooled by Randomness Through Selection Bias" provides an interesting review of selection bias considerations in stock market trading schemes, from the perspective of traders.

Sampling Distribution of a Statistic

The term *sampling distribution* of a statistic refers to the distribution of some sample statistic, over many samples drawn from the same population. Much of classical statistics is concerned with making inferences from (small) samples to (very large) populations.

KEY TERMS

Sample statistic

A metric calculated for a sample of data drawn from a larger population.

Data distribution

The frequency distribution of individual *values* in a data set.

Sampling distribution

The frequency distribution of a *sample statistic* over many samples or resamples.

Central limit theorem

The tendency of the sampling distribution to take on a normal shape as sample size rises.

Standard error

The variability (standard deviation) of a sample *statistic* over many samples (not to be confused with *standard deviation*, which, by itself, refers to variability of individual data *values*).

Typically, a sample is drawn with the goal of measuring something (with a *sample statistic*) or modeling something (with a statistical or machine learning model). Since our estimate or model is based on a sample, it might be in error; it might be different if we were to draw a different sample. We are therefore interested in how different it might be—a key concern is *sampling variability*. If we had lots of data, we could draw additional samples and observe the distribution of a sample statistic directly. Typically, we will calculate our estimate or model using as much data as is easily available, so the option of drawing additional samples from the population is not readily available.

> **WARNING**
>
> It is important to distinguish between the distribution of the individual data points, known as *the data distribution*, and the distribution of a sample statistic, known as the *sampling distribution*.

The distribution of a sample statistic such as the mean is likely to be more regular and bell-shaped than the distribution of the data itself. The larger the sample that the statistic is based on, the more this is true. Also, the larger the sample, the narrower the distribution of the sample statistic.

This is illustrated in an example using annual income for loan applicants to Lending Club (see [Link to Come] for a description of the data). Take three samples from this data: a sample of 1,000 values, a sample of 1,000 means of 5 values, and a sample of 1,000 means of 20 values. Then plot a histogram of each sample to produce Figure 2-6.

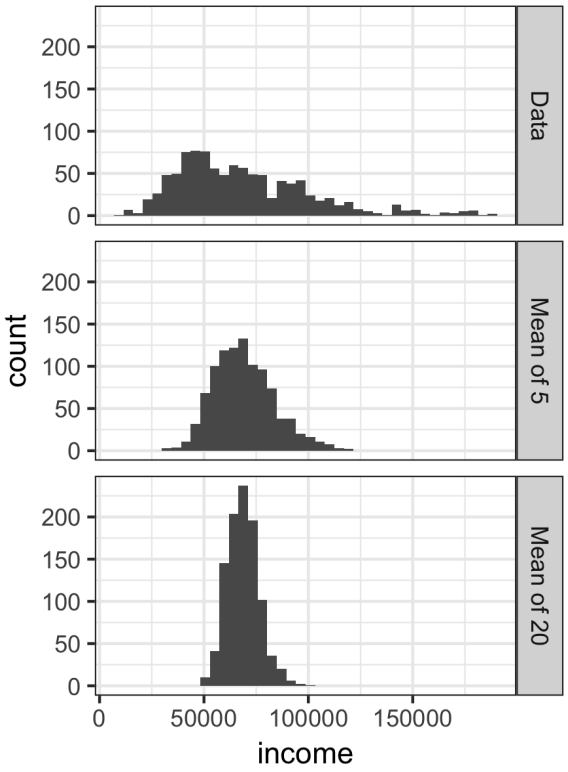

Figure 2-6. Histogram of annual incomes of 1,000 loan applicants (top), then 1000 means of n=5 applicants (middle), and n=20 (bottom)

The histogram of the individual data values is broadly spread out and skewed toward higher values as is to be expected with income data. The histograms of the means of 5 and 20 are increasingly compact and more bell-shaped. Here is the *R* code to generate these histograms, using the visualization package `ggplot2`.

```
library(ggplot2)
# take a simple random sample
samp_data <- data.frame(income=sample(loans_income, 1000),
                        type='data_dist')
# take a sample of means of 5 values
samp_mean_05 <- data.frame(
  income = tapply(sample(loans_income, 1000*5),
                  rep(1:1000, rep(5, 1000)), FUN=mean),
  type = 'mean_of_5')
# take a sample of means of 20 values
samp_mean_20 <- data.frame(
```

```
  income = tapply(sample(loans_income, 1000*20),
                  rep(1:1000, rep(20, 1000)), FUN=mean),
  type = 'mean_of_20')
# bind the data.frames and convert type to a factor
income <- rbind(samp_data, samp_mean_05, samp_mean_20)
income$type = factor(income$type,
                     levels=c('data_dist', 'mean_of_5', 'mean_of_20'),
                     labels=c('Data', 'Mean of 5', 'Mean of 20'))
# plot the histograms
ggplot(income, aes(x=income)) +
  geom_histogram(bins=40) +
  facet_grid(type ~ .)
```

The *Python* code uses `seaborn`'s `FacetGrid` to show the three histograms.

```
import pandas as pd
import seaborn as sns

sample_data = pd.DataFrame({
    'income': loans_income.sample(1000),
    'type': 'Data',
})
sample_mean_05 = pd.DataFrame({
    'income': [loans_income.sample(5).mean() for _ in range(1000)],
    'type': 'Mean of 5',
})
sample_mean_20 = pd.DataFrame({
    'income': [loans_income.sample(20).mean() for _ in range(1000)],
    'type': 'Mean of 20',
})
results = pd.concat([sample_data, sample_mean_05, sample_mean_20])

g = sns.FacetGrid(results, col='type', col_wrap=1, height=2, aspect=2)
g.map(plt.hist, 'income', range=[0, 200000], bins=40)
g.set_axis_labels('Income', 'Count')
g.set_titles('{col_name}')
```

Central Limit Theorem

This phenomenon is termed the *central limit theorem*. It says that the means drawn from multiple samples will resemble the familiar bell-shaped normal curve (see "Normal Distribution"), even if the source population is not normally distributed, provided that the sample size is large enough and the departure of the data from normality is not too great. The central limit theorem allows normal-approximation formulas like the t-distribution to be used in calculating sampling distributions for inference—that is, confidence intervals and hypothesis tests.

The central limit theorem receives a lot of attention in traditional statistics texts because it underlies the machinery of hypothesis tests and confidence intervals, which themselves consume half the space in such texts. Data scientists should be aware of this role, but, since formal hypothesis tests and confidence intervals play a small role in data science, and the bootstrap is available in any case, the central limit theorem is not so central in the practice of data science.

Standard Error

The *standard error* is a single metric that sums up the variability in the sampling distribution for a statistic. The standard error can be estimated using a statistic based on the standard deviation *s* of the sample values, and the sample size *n*:

$$\text{Standard error} = SE = \frac{s}{\sqrt{n}}$$

As the sample size increases, the standard error decreases, corresponding to what was observed in Figure 2-6. The relationship between standard error and sample size is sometimes referred to as the *square-root of n* rule: in order to reduce the standard error by a factor of 2, the sample size must be increased by a factor of 4.

The validity of the standard error formula arises from the central limit theorem (see "Central Limit Theorem"). In fact, you don't need to rely on the central limit theorem to understand standard error. Consider the following approach to measure standard error:

1. Collect a number of brand new samples from the population.
2. For each new sample, calculate the statistic (e.g., mean).
3. Calculate the standard deviation of the statistics computed in step 2; use this as your estimate of standard error.

In practice, this approach of collecting new samples to estimate the standard error is typically not feasible (and statistically very wasteful). Fortunately, it turns out that it is not necessary to draw brand new samples; instead, you can use *bootstrap* resamples (see "The Bootstrap"). In modern statistics, the bootstrap has become the standard way to to estimate standard error. It can be used for virtually any statistic and does not rely on the central limit theorem or other distributional assumptions.

STANDARD DEVIATION VERSUS STANDARD ERROR

Do not confuse standard deviation (which measures the variability of individual data points) with standard error (which measures the variability of a sample metric).

KEY IDEAS

- The frequency distribution of a sample statistic tells us how that metric would turn out differently from sample to sample.
- This sampling distribution can be estimated via the bootstrap, or via formulas that rely on the central limit theorem.
- A key metric that sums up the variability of a sample statistic is its standard error.

Further Reading

David Lane's online multimedia resource in statistics has a useful simulation that allows you to select a sample statistic, a sample size and number of iterations and visualize a histogram of the resulting frequency distribution.

The Bootstrap

One easy and effective way to estimate the sampling distribution of a statistic, or of model parameters, is to draw additional samples, with replacement, from the sample itself and recalculate the statistic or model for each resample. This procedure is called the *bootstrap*, and it does not necessarily involve any assumptions about the data or the sample statistic being normally distributed.

KEY TERMS

Bootstrap sample

A sample taken with replacement from an observed data set.

Resampling

The process of taking repeated samples from observed data; includes both bootstrap and permutation (shuffling) procedures.

Conceptually, you can imagine the bootstrap as replicating the original sample thousands or millions of times so that you have a hypothetical population that embodies all the knowledge from your original sample (it's just larger). You can then draw samples from this hypothetical population for the purpose of estimating a sampling distribution. See Figure 2-7.

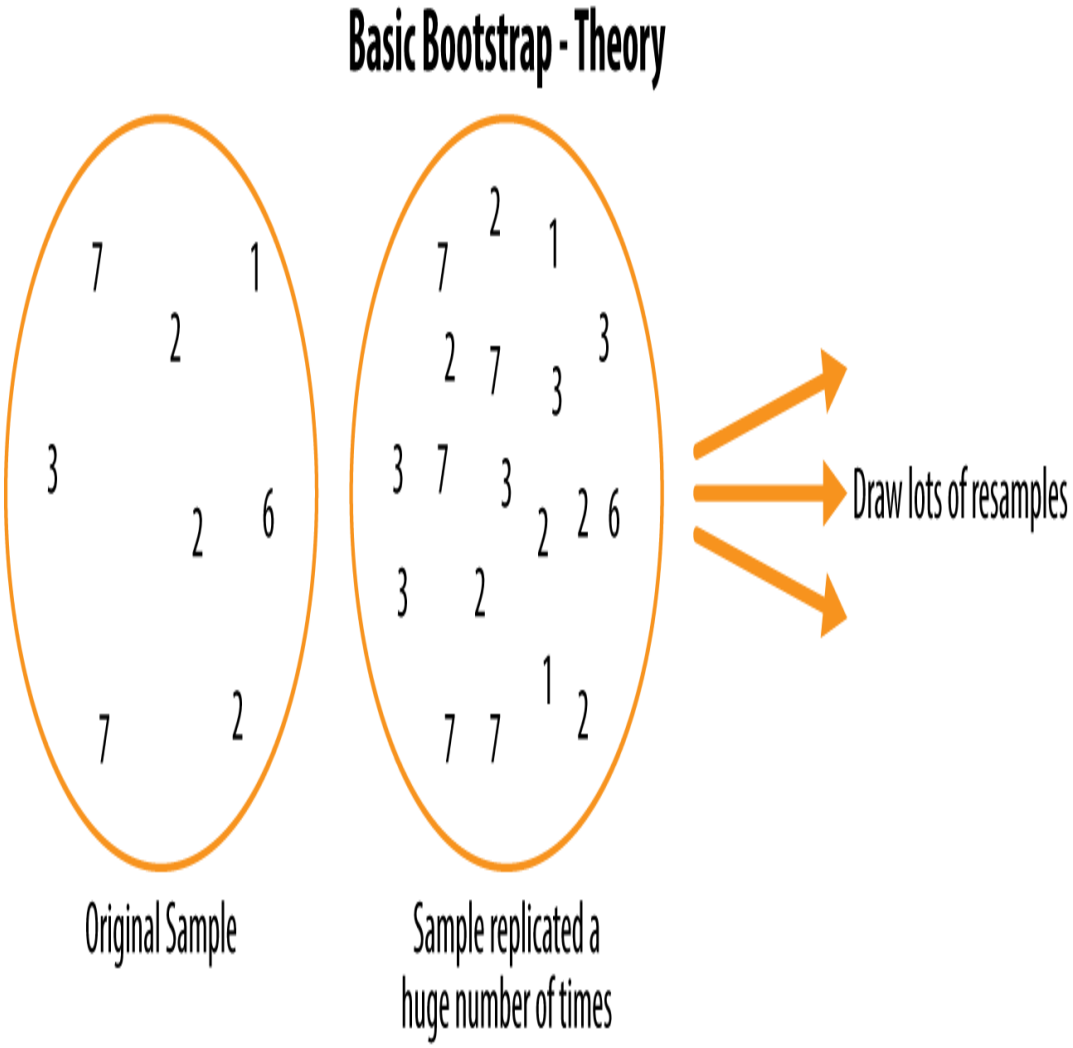

Figure 2-7. The idea of the bootstrap

In practice, it is not necessary to actually replicate the sample a huge number of times. We simply replace each observation after each draw; that is, we *sample with replacement*. In this way we effectively create an infinite population in which the probability of an element being drawn remains unchanged from draw to draw. The algorithm for a bootstrap resampling of the mean is as follows, for a sample of size *n*:

1. Draw a sample value, record, replace it.

2. Repeat *n* times.
3. Record the mean of the *n* resampled values.
4. Repeat steps 1–3 *R* times.
5. Use the *R* results to:
    a. Calculate their standard deviation (this estimates sample mean standard error).
    b. Produce a histogram or boxplot.
    c. Find a confidence interval.

R, the number of iterations of the bootstrap, is set somewhat arbitrarily. The more iterations you do, the more accurate the estimate of the standard error, or the confidence interval. The result from this procedure is a bootstrap set of sample statistics or estimated model parameters, which you can then examine to see how variable they are.

The *R* package `boot` combines these steps in one function. For example, the following applies the bootstrap to the incomes of people taking out loans:

```
library(boot)
stat_fun <- function(x, idx) median(x[idx])
boot_obj <- boot(loans_income, R=1000, statistic=stat_fun)
```

The function `stat_fun` computes the median for a given sample specified by the index `idx`. The result is as follows:

```
Bootstrap Statistics :
    original   bias    std. error
t1*    62000 -70.5595    209.1515
```

The original estimate of the median is $62,000. The bootstrap distribution indicates that the estimate has a *bias* of about –$70 and a standard error of $209. The results will vary slightly between consecutive runs of the algorithm.

The major packages don't provide implementations of the bootstrap approach. It is however straightforward to implement using the `scikit-learn` method `sklearn.utils.resample`.

```
results = []
for nrepeat in range(1000):
    sample = resample(loans_income)
    results.append(sample.median())
results = pd.Series(results)
print('Bootstrap Statistics:')
print(f'original: {loans_income.median()}')
print(f'bias: {results.mean() - loans_income.median()}')
print(f'std. error: {results.std()}')
```

The bootstrap can be used with multivariate data, where the rows are sampled as units (see Figure 2-8). A model might then be run on the bootstrapped data, for example, to estimate the stability (variability) of model parameters, or to improve predictive power. With classification and regression trees (also called *decision trees*), running multiple trees on bootstrap samples and then averaging their predictions (or, with classification, taking a majority vote) generally performs better than using a single tree. This process is called *bagging* (short for "bootstrap aggregating": see [Link to Come]).

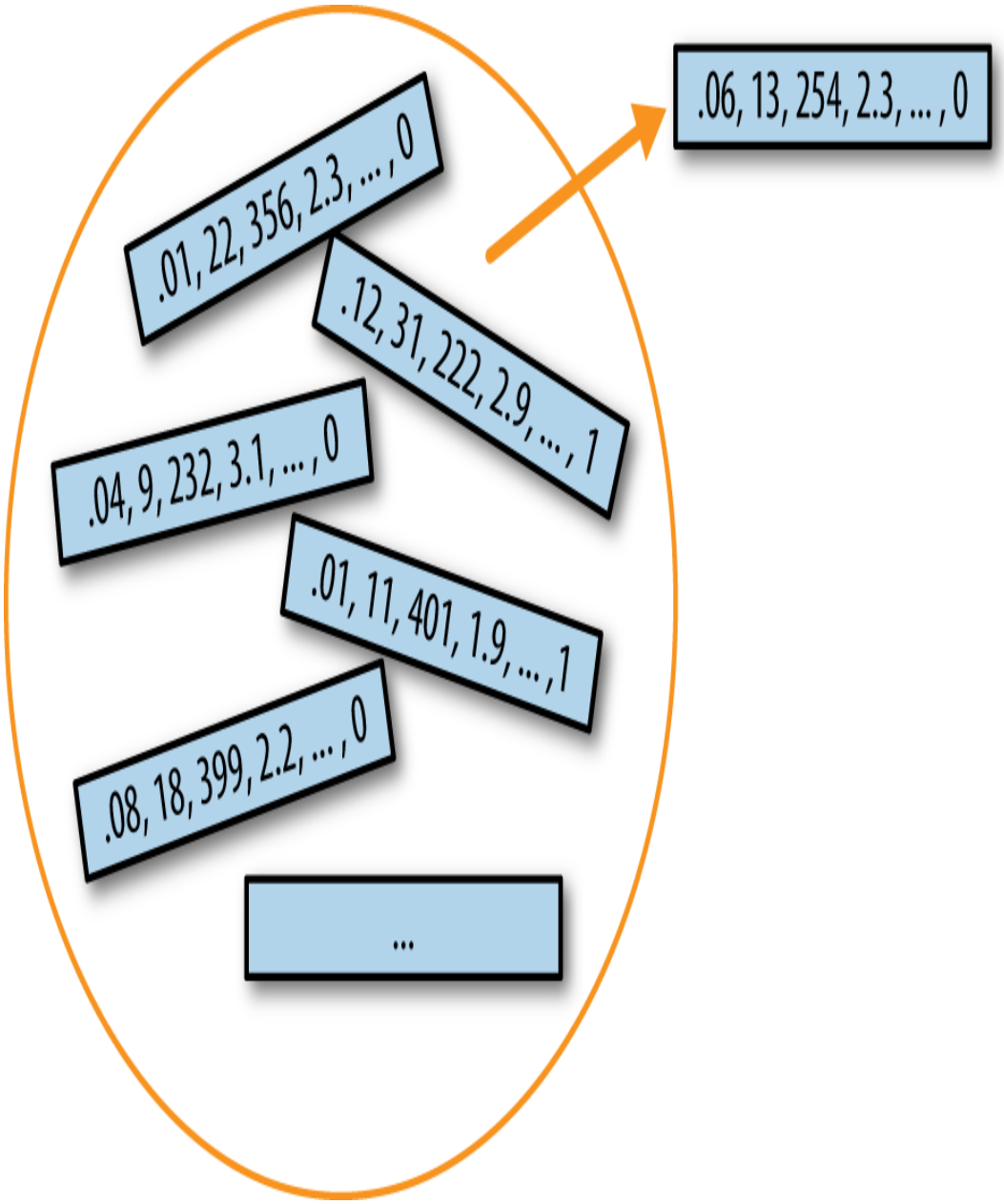

Figure 2-8. Multivariate bootstrap sampling

The repeated resampling of the bootstrap is conceptually simple, and Julian Simon, an economist and demographer, published a compendium of resampling examples, including the bootstrap, in his 1969 text *Basic Research Methods in Social Science* (Random House). However, it is also computationally intensive, and was not a feasible option before the widespread availability of computing

power. The technique gained its name and took off with the publication of several journal articles and a book by Stanford statistician Bradley Efron in the late 1970s and early 1980s. It was particularly popular among researchers who use statistics but are not statisticians, and for use with metrics or models where mathematical approximations are not readily available. The sampling distribution of the mean has been well established since 1908; the sampling distribution of many other metrics has not. The bootstrap can be used for sample size determination; experiment with different values for *n* to see how the sampling distribution is affected.

The bootstrap met with considerable skepticism when it was first introduced; it had the aura to many of spinning gold from straw. This skepticism stemmed from a misunderstanding of the bootstrap's purpose.

> **WARNING**
>
> The bootstrap does not compensate for a small sample size; it does not create new data, nor does it fill in holes in an existing data set. It merely informs us about how lots of additional samples would behave when drawn from a population like our original sample.

Resampling versus Bootstrapping

Sometimes the term *resampling* is used synonymously with the term *bootstrapping*, as just outlined. More often, the term *resampling* also includes permutation procedures (see [Link to Come]), where multiple samples are combined and the sampling may be done

without replacement. In any case, the term *bootstrap* always implies sampling with replacement from an observed data set.

KEY IDEAS

- The bootstrap (sampling with replacement from a data set) is a powerful tool for assessing the variability of a sample statistic.
- The bootstrap can be applied in similar fashion in a wide variety of circumstances, without extensive study of mathematical approximations to sampling distributions.
- It also allows us to estimate sampling distributions for statistics where no mathematical approximation has been developed.
- When applied to predictive models, aggregating multiple bootstrap sample predictions (bagging) outperforms the use of a single model.

Further Reading

- *An Introduction to the Bootstrap* by Bradley Efron and Robert Tibshirani (Chapman Hall, 1993) was the first book-length treatment of the bootstrap. It is still widely read.
- The retrospective on the bootstrap in the May 2003 issue of *Statistical Science*, (vol. 18, no. 2), discusses (among other antecedents, in Peter Hall's "Prehistory") Julian Simon's first publication of the bootstrap in 1969.
- See *An Introduction to Statistical Learning* by Gareth James et al. (Springer, 2013) for sections on the bootstrap and, in particular, bagging.

Confidence Intervals

Frequency tables, histograms, boxplots, and standard errors are all ways to understand the potential error in a sample estimate. Confidence intervals are another.

KEY TERMS

Confidence level

The percentage of confidence intervals, constructed in the same way from the same population, expected to contain the statistic of interest.

Interval endpoints

The top and bottom of the confidence interval.

There is a natural human aversion to uncertainty; people (especially experts) say, "I don't know" far too rarely. Analysts and managers, while acknowledging uncertainty, nonetheless place undue faith in an estimate when it is presented as a single number (a *point estimate*). Presenting an estimate not as a single number but as a range is one way to counteract this tendency. Confidence intervals do this in a manner grounded in statistical sampling principles.

Confidence intervals always come with a coverage level, expressed as a (high) percentage, say 90% or 95%. One way to think of a 90% confidence interval is as follows: it is the interval that encloses the central 90% of the bootstrap sampling distribution of a sample statistic (see "The Bootstrap"). More generally, an *x*% confidence interval around a sample estimate should, on average, contain similar sample estimates *x*% of the time (when a similar sampling procedure is followed).

Given a sample of size *n*, and a sample statistic of interest, the algorithm for a bootstrap confidence interval is as follows:

1. Draw a random sample of size *n* with replacement from the data (a resample).

2. Record the statistic of interest for the resample.
3. Repeat steps 1–2 many (*R*) times.
4. For an *x*% confidence interval, trim [(100-*x*) / 2]% of the *R* resample results from either end of the distribution.
5. The trim points are the endpoints of an *x*% bootstrap confidence interval.

Figure 2-9 shows a a 90% confidence interval for the mean annual income of loan applicants, based on a sample of 20 for which the mean was $57,573.

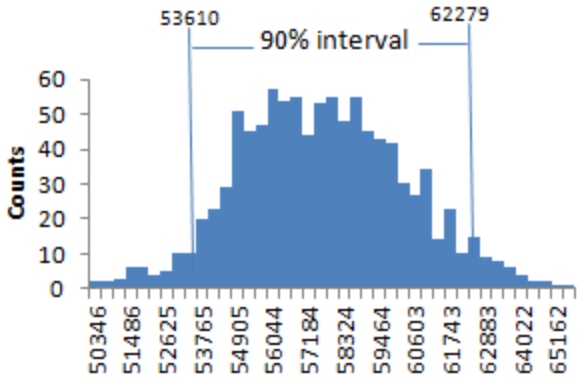

Figure 2-9. Bootstrap confidence interval for the annual income of loan applicants, based on a sample of 20

The bootstrap is a general tool that can be used to generate confidence intervals for most statistics, or model parameters. Statistical textbooks and software, with roots in over a half-century of computerless statistical analysis, will also reference confidence intervals generated by formulas, especially the t-distribution (see "Student's t-Distribution").

NOTE

Of course, what we are really interested in when we have a sample result is "what is the probability that the true value lies within a certain interval?" This is not really the question that a confidence interval answers, but it ends up being how most people interpret the answer.

The probability question associated with a confidence interval starts out with the phrase "Given a sampling procedure and a population, what is the probability that…" To go in the opposite direction, "Given a sample result, what is the probability that (something is true about the population)," involves more complex calculations and deeper imponderables.

The percentage associated with the confidence interval is termed the *level of confidence*. The higher the level of confidence, the wider the interval. Also, the smaller the sample, the wider the interval (i.e., the more uncertainty). Both make sense: the more confident you want to be, and the less data you have, the wider you must make the confidence interval to be sufficiently assured of capturing the true value.

NOTE

For a data scientist, a confidence interval is a tool to get an idea of how variable a sample result might be. Data scientists would use this information not to publish a scholarly paper or submit a result to a regulatory agency (as a researcher might), but most likely to communicate the potential error in an estimate, and, perhaps, learn whether a larger sample is needed.

KEY IDEAS

- Confidence intervals are the typical way to present estimates as an interval range.
- The more data you have, the less variable a sample estimate will be.
- The lower the level of confidence you can tolerate, the narrower the confidence interval will be.
- The bootstrap is an effective way to construct confidence intervals.

Further Reading

- For a bootstrap approach to confidence intervals, see *Introductory Statistics and Analytics: A Resampling Perspective* by Peter Bruce (Wiley, 2014) or *Statistics* by Robin Lock and four other Lock family members (Wiley, 2012).
- Engineers, who have a need to understand the precision of their measurements, use confidence intervals perhaps more than most disciplines, and *Modern Engineering Statistics* by Tom Ryan (Wiley, 2007) discusses confidence intervals. It also reviews a tool that is just as useful and gets less attention: prediction intervals (intervals around a single value, as opposed to a mean or other summary statistic).

Normal Distribution

The bell-shaped normal distribution is iconic in traditional statistics.[1] The fact that distributions of sample statistics are often normally shaped has made it a powerful tool in the development of mathematical formulas that approximate those distributions.

KEY TERMS

Error

The difference between a data point and a predicted or average value.

Standardize

Subtract the mean and divide by the standard deviation.

z-score

The result of standardizing an individual data point.

Standard normal

A normal distribution with mean = 0 and standard deviation = 1.

QQ-Plot

A plot to visualize how close a sample distribution is to a normal distribution.

In a normal distribution (Figure 2-10), 68% of the data lies within one standard deviation of the mean, and 95% lies within two standard deviations.

WARNING

It is a common misconception that the normal distribution is called that because most data follows a normal distribution—that is, it is the normal thing. Most of the variables used in a typical data science project—in fact most raw data as a whole—are *not* normally distributed: see "Long-Tailed Distributions". The utility of the normal distribution derives from the fact that many statistics *are* normally distributed in their sampling distribution. Even so, assumptions of normality are generally a last resort, used when empirical probability distributions, or bootstrap distributions, are not available.

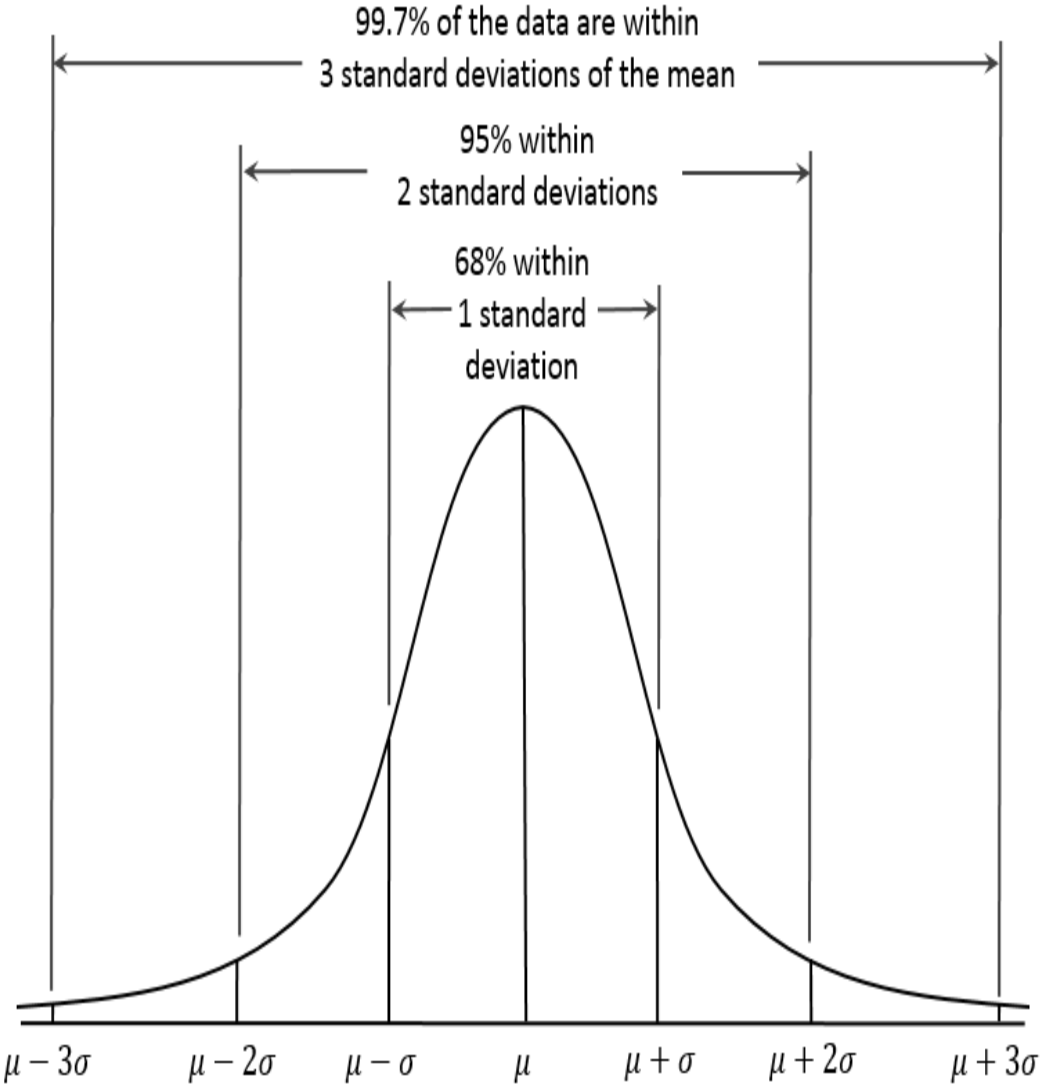

Figure 2-10. Normal curve

NOTE

The normal distribution is also referred to as a *Gaussian* distribution after Carl Friedrich Gauss, a prodigious German mathematician from the late 18th and early 19th century. Another name previously used for the normal distribution was the "error" distribution. Statistically speaking, an *error* is the difference between an actual value and a statistical estimate like the sample mean. For example, the standard deviation (see "Estimates of Variability") is based on the errors from the mean of the data. Gauss's development of the normal distribution came from his study of the errors of astronomical measurements that were found to be normally distributed.

Standard Normal and QQ-Plots

A *standard normal* distribution is one in which the units on the x-axis are expressed in terms of standard deviations away from the mean. To compare data to a standard normal distribution, you subtract the mean then divide by the standard deviation; this is also called *normalization* or *standardization* (see [Link to Come]). Note that "standardization" in this sense is unrelated to database record standardization (conversion to a common format). The transformed value is termed a *z-score*, and the normal distribution is sometimes called the *z-distribution*.

A QQ-Plot is used to visually determine how close a sample is to a specified distribution, in this case the normal. The QQ-Plot orders the *z*-scores from low to high, and plots each value's *z*-score on the y-axis; the x-axis is the corresponding quantile of a normal distribution for that value's rank. Since the data is normalized, the units correspond to the number of standard deviations away of the data from the mean. If the points roughly fall on the diagonal line, then the

sample distribution can be considered close to normal. Figure 2-11 shows a QQ-Plot for a sample of 100 values randomly generated from a normal distribution; as expected, the points closely follow the line. This figure can be produced in *R* with the `qqnorm` function:

```
norm_samp <- rnorm(100)
qqnorm(norm_samp)
abline(a=0, b=1, col='grey')
```

In *Python*, use the method `scipy.stats.probplot` to create the QQ-Plot.

```
fig, ax = plt.subplots(figsize=(4, 4))
norm_sample = stats.norm.rvs(size=100)
stats.probplot(norm_sample, plot=ax)
```

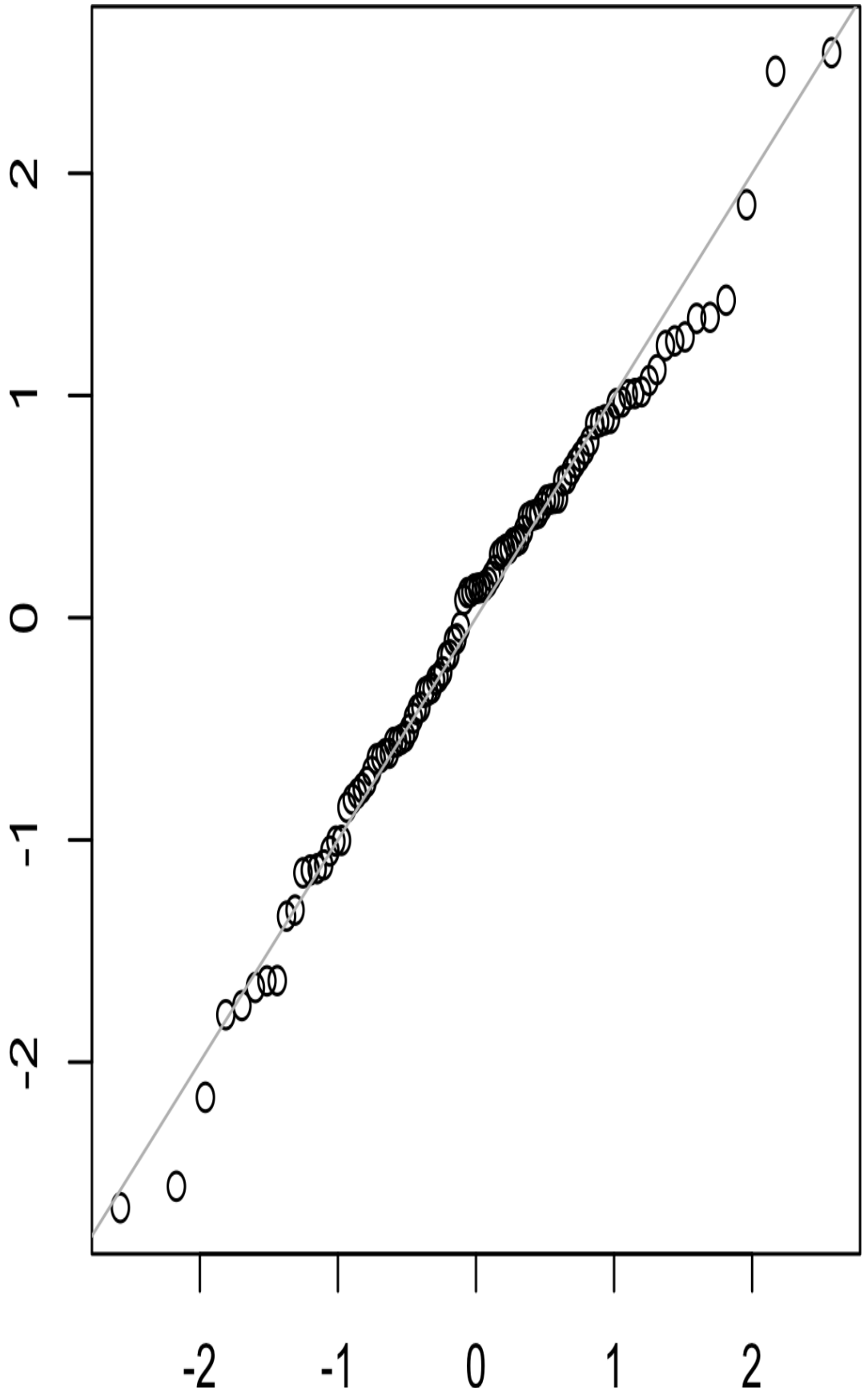
2
1
0
-1
-2
-2
-1
0
1
2

Figure 2-11. QQ-Plot of a sample of 100 values drawn from a standard normal distribution

WARNING

Converting data to *z*-scores (i.e., standardizing or normalizing the data) does *not* make the data normally distributed. It just puts the data on the same scale as the standard normal distribution, often for comparison purposes.

KEY IDEAS

- The normal distribution was essential to the historical development of statistics, as it permitted mathematical approximation of uncertainty and variability.
- While raw data is typically not normally distributed, errors often are, as are averages and totals in large samples.
- To convert data to *z*-scores, you subtract the mean of the data and divide by the standard deviation; you can then compare the data to a normal distribution.

Long-Tailed Distributions

Despite the importance of the normal distribution historically in statistics, and in contrast to what the name would suggest, data is generally not normally distributed.

KEY TERMS FOR LONG-TAIL DISTRIBUTION

Tail

The long narrow portion of a frequency distribution, where relatively extreme values occur at low frequency.

Skew

Where one tail of a distribution is longer than the other.

While the normal distribution is often appropriate and useful with respect to the distribution of errors and sample statistics, it typically does not characterize the distribution of raw data. Sometimes, the distribution is highly *skewed* (asymmetric), such as with income data, or the distribution can be discrete, as with binomial data. Both symmetric and asymmetric distributions may have *long tails*. The tails of a distribution correspond to the extreme values (small and large). Long tails, and guarding against them, are widely recognized in practical work. Nassim Taleb has proposed the *black swan* theory, which predicts that anomalous events, such as a stock market crash, are much more likely to occur than would be predicted by the normal distribution.

A good example to illustrate the long-tailed nature of data is stock returns. Figure 2-12 shows the QQ-Plot for the daily stock returns for Netflix (NFLX). This is generated in *R* by:

```
nflx <- sp500_px[,'NFLX']
nflx <- diff(log(nflx[nflx>0]))
qqnorm(nflx)
abline(a=0, b=1, col='grey')
```

The corresponding *Python* code is:

```
nflx = sp500_px.NFLX
nflx = np.diff(np.log(nflx[nflx>0]))
fig, ax = plt.subplots(figsize=(4, 4))
stats.probplot(nflx, plot=ax)
```

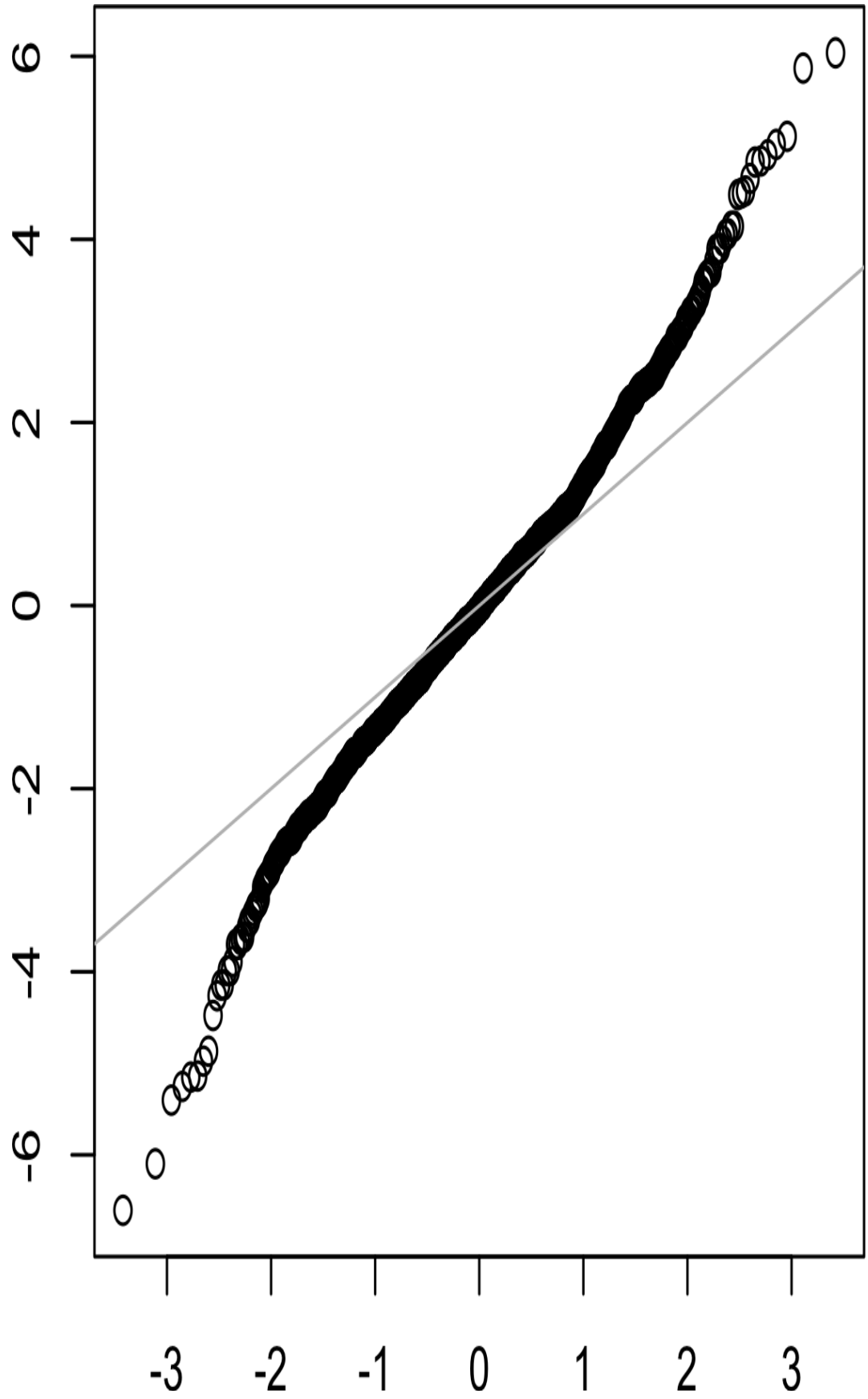
6
4
2
0
-2
-4
-6
-3
-2
-1
0
1
2
3

Figure 2-12. QQ-Plot of the returns for NFLX

In contrast to Figure 2-11, the points are far below the line for low values and far above the line for high values. This means that we are much more likely to observe extreme values than would be expected if the data had a normal distribution. Figure 2-12 shows another common phenomena: the points are close to the line for the data within one standard deviation of the mean. Tukey refers to this phenomenon as data being "normal in the middle," but having much longer tails (see [Tukey-1987]).

> **NOTE**
>
> There is much statistical literature about the task of fitting statistical distributions to observed data. Beware an excessively data-centric approach to this job, which is as much art as science. Data is variable, and often consistent, on its face, with more than one shape and type of distribution. It is typically the case that domain and statistical knowledge must be brought to bear to determine what type of distribution is appropriate to model a given situation. For example, we might have data on the level of internet traffic on a server over many consecutive 5-second periods. It is useful to know that the best distribution to model "events per time period" is the Poisson (see "Poisson Distributions").

> **KEY IDEAS FOR LONG-TAIL DISTRIBUTION**
>
> - Most data is not normally distributed.
> - Assuming a normal distribution can lead to underestimation of extreme events ("black swans").

Further Reading

- *The Black Swan*, 2nd ed., by Nassim Taleb (Random House, 2010).
- *Handbook of Statistical Distributions with Applications*, 2nd ed., by K. Krishnamoorthy (CRC Press, 2016)

Student's t-Distribution

The *t-distribution* is a normally shaped distribution, but a bit thicker and longer on the tails. It is used extensively in depicting distributions of sample statistics. Distributions of sample means are typically shaped like a t-distribution, and there is a family of t-distributions that differ depending on how large the sample is. The larger the sample, the more normally shaped the t-distribution becomes.

KEY TERMS FOR STUDENT'S T-DISTRIBUTION

n

Sample size.

Degrees of freedom

A parameter that allows the t-distribution to adjust to different sample sizes, statistics, and number of groups.

The t-distribution is often called *Student's t* because it was published in 1908 in *Biometrika* by W. S. Gosset under the name "Student." Gosset's employer, the Guinness brewery, did not want competitors to know that it was using statistical methods, so insisted that Gosset not use his name on the article.

Gosset wanted to answer the question “What is the sampling distribution of the mean of a sample, drawn from a larger population?” He started out with a resampling experiment—drawing random samples of 4 from a data set of 3,000 measurements of criminals’ height and left-middle-finger lengths. (This being the era of eugenics, there was much interest in data on criminals, and in discovering correlations between criminal tendencies and physical or psychological attributes.) He plotted the standardized results (the *z*-scores) on the x-axis and the frequency on the y-axis. Separately, he had derived a function, now known as *Student’s t*, and he fit this function over the sample results, plotting the comparison (see Figure 2-13).

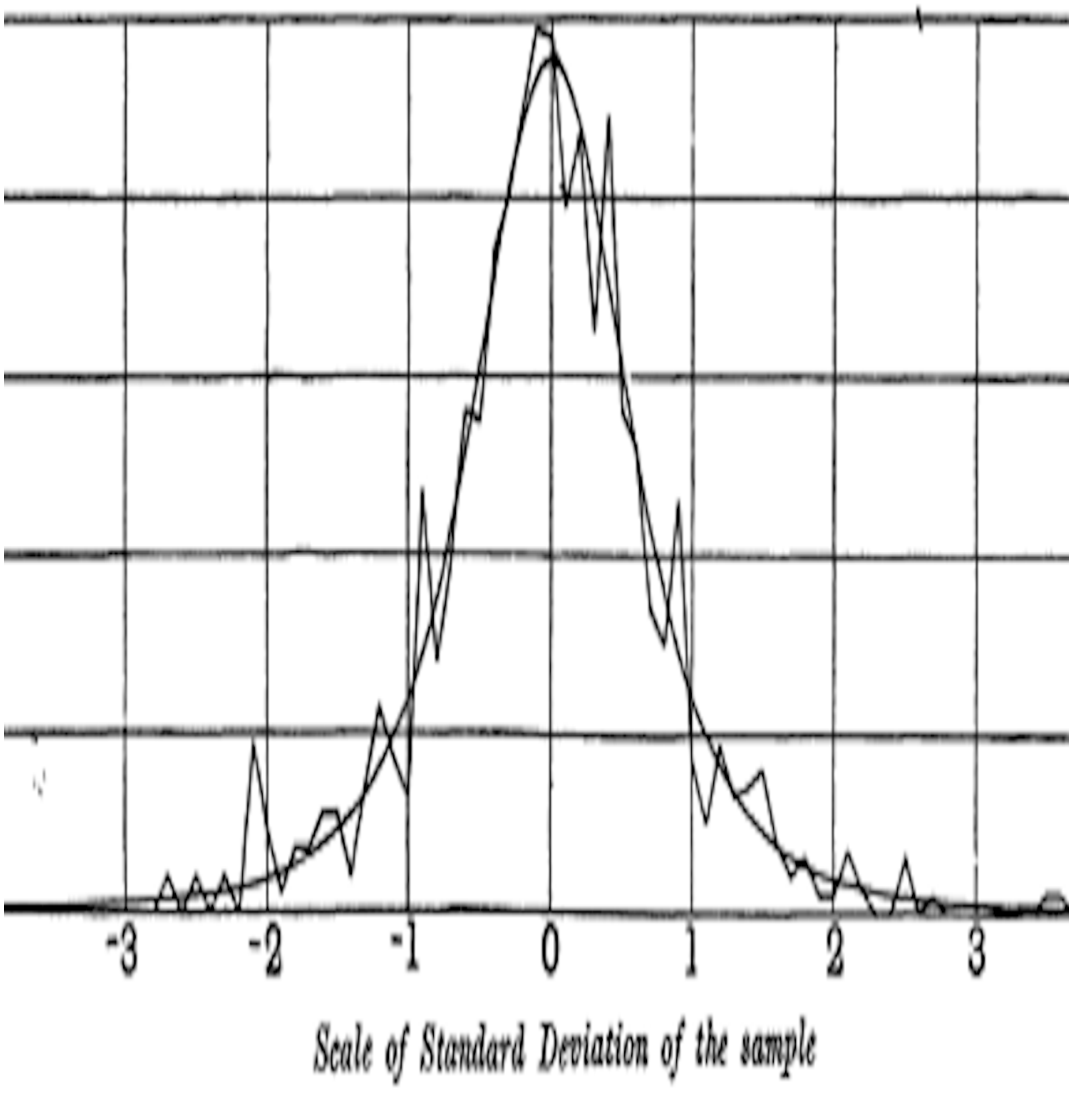

Figure 2-13. Gosset's resampling experiment results and fitted t-curve (from his 1908 Biometrika paper)

A number of different statistics can be compared, after standardization, to the t-distribution, to estimate confidence intervals in light of sampling variation. Consider a sample of size *n* for which the sample mean $\bar{x}$ has been calculated. If *s* is the sample standard deviation, a 90% confidence interval around the sample mean is given by:

$$\bar{x} \pm t_{n-1}\left(0.05\right) \cdot \frac{s}{\sqrt{n}}$$

where $t_{n-1}\left(.05\right)$ is the value of the t-statistic, with (n – 1) degrees of freedom (see [Link to Come]), that "chops off" 5% of the t-distribution at either end. The t-distribution has been used as a reference for the distribution of a sample mean, the difference between two sample means, regression parameters, and other statistics.

Had computing power been widely available in 1908, statistics would no doubt have relied much more heavily on computationally intensive resampling methods from the start. Lacking computers, statisticians turned to mathematics and functions such as the t-distribution to approximate sampling distributions. Computer power enabled practical resampling experiments in the 1980s, but by then, use of the t-distribution and similar distributions had become deeply embedded in textbooks and software.

The t-distribution's accuracy in depicting the behavior of a sample statistic requires that the distribution of that statistic for that sample be shaped like a normal distribution. It turns out that sample statistics *are* often normally distributed, even when the underlying population data is not (a fact which led to widespread application of the t-distribution). This phenomenon is termed the *central limit theorem* (see "Central Limit Theorem").

NOTE

What do data scientists need to know about the t-distribution and the central limit theorem? Not a whole lot. These distributions are used in classical statistical inference, but are not as central to the purposes of data science. Understanding and quantifying uncertainty and variation are important to data scientists, but empirical bootstrap sampling can answer most questions about sampling error. However, data scientists will routinely encounter t-statistics in output from statistical software and statistical procedures in *R*, for example in A-B tests and regressions, so familiarity with its purpose is helpful.

KEY IDEAS

- The t-distribution is actually a family of distributions resembling the normal distribution, but with thicker tails.
- It is widely used as a reference basis for the distribution of sample means, differerences between two sample means, regression parameters, and more.

Further Reading

- The original Gosset paper in *Biometrika* from 1908 is available as a PDF.
- A standard treatment of the t-distribution can be found in David Lane's online resource.

Binomial Distribution

KEY TERMS FOR BINOMIAL DISTRIBUTION

Trial

An event with a discrete outcome (e.g., a coin flip).

Success

The outcome of interest for a trial.

Synonyms

"1" (as opposed to "0")

Binomial

Having two outcomes.

Synonyms

yes/no, 0/1, binary

Binomial trial

A trial with two outcomes.

Synonym

Bernoulli trial

Binomial distribution

Distribution of number of successes in *x* trials.

Synonym

Bernoulli distribution

Yes/no (binomial) outcomes lie at the heart of analytics since they are often the culmination of a decision or other process; buy/don't buy, click/don't click, survive/die, and so on. Central to understanding the binomial distribution is the idea of a set of *trials*, each trial having two possible outcomes with definite probabilities.

For example, flipping a coin 10 times is a binomial experiment with 10 trials, each trial having two possible outcomes (heads or tails); see

Figure 2-14. Such yes/no or 0/1 outcomes are termed *binary* outcomes, and they need not have 50/50 probabilities. Any probabilities that sum to 1.0 are possible. It is conventional in statistics to term the "1" outcome the *success* outcome; it is also common practice to assign "1" to the more rare outcome. Use of the term *success* does not imply that the outcome is desirable or beneficial, but it does tend to indicate the outcome of interest. For example, loan defaults or fraudulent transactions are relatively uncommon events that we may be interested in predicting, so they are termed "1s" or "successes."

Figure 2-14. The tails side of a buffalo nickel

The binomial distribution is the frequency distribution of the number of successes (*x*) in a given number of trials (*n*) with specified probability (*p*) of success in each trial. There is a family of binomial distributions, depending on the values of *n*, and *p*. The binomial distribution would answer a question like:

> *If the probability of a click converting to a sale is 0.02, what is the probability of observing 0 sales in 200 clicks?*

The *R* function `dbinom` calculates binomial probabilities. For example:

```
dbinom(x=2, size=5, p=0.1)
```

would return 0.0729, the probability of observing exactly $x = 2$ successes in $n = 5$ trials, where the probability of success for each trial is $p = 0.1$.

Often we are interested in determining the probability of x or fewer successes in n trials. In this case, we use the function `pbinom`:

```
pbinom(2, 5, 0.1)
```

This would return 0.9914, the probability of observing two or fewer successes in five trials, where the probability of success for each trial is 0.1.

The `scipy.stats` module implements a large variety of statistical distributions. For the binomial distribution use the functions `stats.binom.pmf` and `stats.binom.cdf`.

```
stats.binom.pmf(2, n=5, p=0.1)
stats.binom.cdf(2, n=5, p=0.1)
```

The mean of a binomial distribution is $n \times p$; you can also think of this as the expected number of successes in n trials, for success probability = p.

The variance is $n \times p(1 - p)$. With a large enough number of trials (particularly when p is close to 0.50), the binomial distribution is virtually indistinguishable from the normal distribution. In fact, calculating binomial probabilities with large sample sizes is computationally demanding, and most statistical procedures use the normal distribution, with mean and variance, as an approximation.

KEY IDEAS

- Binomial outcomes are important to model, since they represent, among other things, fundamental decisions (buy or don't buy, click or don't click, survive or die, etc.).
- A binomial trial is an experiment with two possible outcomes: one with probability *p* and the other with probability *1 – p*.
- With large *n*, and provided *p* is not too close to 0 or 1, the binomial distribution can be approximated by the normal distribution.

Further Reading

- Read about the "quincunx", a pinball-like simulation device for illustrating the binomial distribution.
- The binomial distribution is a staple of introductory statistics, and all introductory statistics texts will have a chapter or two on it.

Chi Square Distribution

An important idea in statistics is *departure from expectation*, especially with respect to category counts. For example, you might want to test whether one variable (say a row variable representing gender) is independent of another (say a column variable representing "was promoted in job"), and you have counts of the number in each of the cells of the data table. The statistic that measures the extent to which results depart from the null expectation of independence is the chi-square statistic. It is the difference between the observed and expected values, divided by the square root of the expected value, squared, then summed across all categories. A more general way of putting this is to note that the chi-square statistic is a measure of the

extent to which a set of observed values “fits” a specified distribution (a “goodness-of-fit” test).

The chi-square distribution is the distribution of this statistic under repeated resampled draws from the null model - see [Link to Come] for a detailed algorithm, and the chi-square formula for a data table. A low chi-square value for a set of counts indicates that they closely follow the expected distribution. A high chi-square indicates that they differ markedly from what is expected.

KEY IDEAS

- The chi-square distribution is typically concerned with counts of subjects or items falling into categories.
- The chi-square statistic measures the extent of departure from what you would expect in a null model.

Further Reading

- The chi-square distribution owes its place in modern statistics to the great statistician Karl Pearson and the birth of hypothesis testing - read about this and more in David Salsburg’s *The Lady Tasting Tea: How Statistics Revolutionized Science in the Twentieth Century* (W. H. Freeman, 2001).
- For a more detailed exposition, see the section in this book on the chi-square test ([Link to Come])

[[F dist]] === F Distribution

A common procedure in scientific experimentation is to test multiple treatments across groups, say different fertilizers on different blocks

of a field. This is similar to the A-B-C test referred to in the chi-square distribution (see "Chi Square Distribution", except we are dealing with measured continuous values rather than counts. In such a case we are interested in the extent to which differences among group means are greater than we might expect under normal random variation. The F-statistic measures this, and is the ratio of the variability among the group means to the variability within each group (also called residual variability). This is termed an *analysis of variance* (see ANOVA, [Link to Come]). The distribution of the F-statistic is the frequency distribution of all the values that would be produced by randomly permuting data in which all the group means are equal (i.e. a null model). There are a variety of F distributions associated with different degrees-of-freedom (e.g. numbers of groups, see [Link to Come]). The calculation of F is illustrated in the section on ANOVA (see [Link to Come]). The F statistic is also used in linear regression to compare the variation accounted for by the regression model to the overall variation in the data. F-statistics are produced automatically by *R* and *Python* as part of regression and ANOVA routines.

KEY IDEAS

- The F distribution is used with experiments and linear models involving measured data.
- The F-statistic compares variation due to factors of interest to overall variation.

Further Reading

- George Cobb's *Introduction to Design and Analysis of Experiments* (Wiley, 2008) contains an excellent exposition

of the decomposition of variance components, which helps understand ANOVA and the F-statistic.

Poisson and Related Distributions

Many processes produce events randomly at a given overall rate—visitors arriving at a website, cars arriving at a toll plaza (events spread over time), imperfections in a square meter of fabric, or typos per 100 lines of code (events spread over space).

KEY TERMS FOR POISSON AND RELATED DISTRIBUTIONS

Lambda

The rate (per unit of time or space) at which events occur.

Poisson distribution

The frequency distribution of the number of events in sampled units of time or space.

Exponential distribution

The frequency distribution of the time or distance from one event to the next event.

Weibull distribution

A generalized version of the exponential, in which the event rate is allowed to shift over time.

Poisson Distributions

From prior data we can estimate the average number of events per unit of time or space, but we might also want to know how different this might be from one unit of time/space to another. The Poisson distribution tells us the distribution of events per unit of time or space when we sample many such units. It is useful when addressing queuing questions like “How much capacity do we need to be 95%

sure of fully processing the internet traffic that arrives on a server in any 5-second period?”

The key parameter in a Poisson distribution is λ, or lambda. This is the mean number of events that occurs in a specified interval of time or space. The variance for a Poisson distribution is also λ.

A common technique is to generate random numbers from a Poisson distribution as part of a queuing simulation. The `rpois` function in *R* does this, taking only two arguments—the quantity of random numbers sought, and lambda:

```
rpois(100, lambda=2)
```

The corresponding `SciPy` function is `stats.poisson.rvs`.

```
stats.poisson.rvs(2, size=100)
```

This code will generate 100 random numbers from a Poisson distribution with $\lambda = 2$. For example, if incoming customer service calls average 2 per minute, this code will simulate 100 minutes, returning the number of calls in each of those 100 minutes.

Exponential Distribution

Using the same parameter λ that we used in the Poisson distribution, we can also model the distribution of the time between events: time between visits to a website or between cars arriving at a toll plaza. It is also used in engineering to model time to failure, and in process management to model, for example, the time required per service

call. The *R* code to generate random numbers from an exponential distribution takes two arguments, *n* (the quantity of numbers to be generated), and *rate*, the number of events per time period. For example:

```
rexp(n=100, rate=0.2)
```

In the function `stats.expon.rvs`, the order of the arguments is reversed.

```
stats.expon.rvs(0.2, size=100)
```

This code would generate 100 random numbers from an exponential distribution where the mean number of events per time period is 0.2. So you could use it to simulate 100 intervals, in minutes, between service calls, where the average rate of incoming calls is 0.2 per minute.

A key assumption in any simulation study for either the Poisson or exponential distribution is that the rate, λ, remains constant over the period being considered. This is rarely reasonable in a global sense; for example, traffic on roads or data networks varies by time of day and day of week. However, the time periods, or areas of space, can usually be divided into segments that are sufficiently homogeneous so that analysis or simulation within those periods is valid.

Estimating the Failure Rate

In many applications, the event rate, λ, is known or can be estimated from prior data. However, for rare events, this is not necessarily so.

Aircraft engine failure, for example, is sufficiently rare (thankfully) that, for a given engine type, there may be little data on which to base an estimate of time between failures. With no data at all, there is little basis on which to estimate an event rate. However, you can make some guesses: if no events have been seen after 20 hours, you can be pretty sure that the rate is not 1 per hour. Via simulation, or direct calculation of probabilities, you can assess different hypothetical event rates and estimate threshold values below which the rate is very unlikely to fall. If there is some data but not enough to provide a precise, reliable estimate of the rate, a goodness-of-fit test (see [Link to Come]) can be applied to various rates to determine how well they fit the observed data.

Weibull Distribution

In many cases, the event rate does not remain constant over time. If the period over which it changes is much longer than the typical interval between events, there is no problem; you just subdivide the analysis into the segments where rates are relatively constant, as mentioned before. If, however, the event rate changes over the time of the interval, the exponential (or Poisson) distributions are no longer useful. This is likely to be the case in mechanical failure—the risk of failure increases as time goes by. The *Weibull* distribution is an extension of the exponential distribution, in which the event rate is allowed to change, as specified by a *shape parameter*, β. If $\beta > 1$, the probability of an event increases over time, if $\beta < 1$, it decreases. Because the Weibull distribution is used with time-to-failure analysis instead of event rate, the second parameter is expressed in terms of characteristic life, rather than in terms of the rate of events per

interval. The symbol used is η, the Greek letter eta. It is also called the *scale* parameter.

With the Weibull, the estimation task now includes estimation of both parameters, β and η. Software is used to model the data and yield an estimate of the best-fitting Weibull distribution.

The *R* code to generate random numbers from a Weibull distribution takes three arguments, `n` (the quantity of numbers to be generated), `shape`, and `scale`. For example, the following code would generate 100 random numbers (lifetimes) from a Weibull distribution with shape of 1.5 and characteristic life of 5,000:

```
rweibull(100, 1.5, 5000)
```

To achieve the same in *Python*, use the function `stats.weibull_min.rvs`.

```
stats.weibull_min.rvs(1.5, scale=5000, size=100)
```

KEY IDEAS

- For events that occur at a constant rate, the number of events per unit of time or space can be modeled as a Poisson distribution.
- In this scenario, you can also model the time or distance between one event and the next as an exponential distribution.
- A changing event rate over time (e.g., an increasing probability of device failure) can be modeled with the Weibull distribution.

Further Reading

- *Modern Engineering Statistics* by Tom Ryan (Wiley, 2007) has a chapter devoted to the probability distributions used in engineering applications.
- Read an engineering-based perspective on the use of the Weibull distribution (mainly from an engineering perspective) here and here.

Summary

In the era of big data, the principles of random sampling remain important when accurate estimates are needed. Random selection of data can reduce bias and yield a higher quality data set than would result from just using the conveniently available data. Knowledge of various sampling and data generating distributions allows us to quantify potential errors in an estimate that might be due to random variation. At the same time, the bootstrap (sampling with replacement from an observed data set) is an attractive "one size fits all" method to determine possible error in sample estimates.

---

1 The bell curve is iconic but perhaps overrated. George W. Cobb, the Mount Holyoke statistician noted for his contribution to the philosophy of teaching introductory statistics, argued in a November 2015 editorial in the *American Statistician* that the "standard introductory course, which puts the normal distribution at its center, had outlived the usefulness of its centrality."

About the Authors

Peter Bruce founded and grew the Institute for Statistics Education at Statistics.com, which now offers about 100 courses in statistics, roughly a third of which are aimed at the data scientist. In recruiting top authors as instructors and forging a marketing strategy to reach professional data scientists, Peter has developed both a broad view of the target market and his own expertise to reach it.

Andrew Bruce has over 30 years of experience in statistics and data science in academia, government, and business. He has a PhD in statistics from the University of Washington and has published numerous papers in refereed journals. He has developed statistical-based solutions to a wide range of problems faced by a variety of industries, from established financial firms to internet startups, and offers a deep understanding of the practice of data science.

Peter Gedeck, Senior Data Scientist at Collaborative Drug Discovery, specializes in the development of machine learning algorithms to predict biological and physicochemical properties of drug candidates. Co-author of Data Mining for Business Analytics, he earned PhD’s in Chemistry from the University of Erlangen-Nürnberg in Germany and Mathematics from Fernuniversität Hagen, Germany