Contents

CHAPTER 1

Notation and conventions

A dataset as a collection of d-tuples (a d-tuple is an ordered list of d elements). Tuples differ from vectors, because we can always add and subtract vectors, but we cannot necessarily add or subtract tuples. There are always N items in any dataset. There are always d elements in each tuple in a dataset. The number of elements will be the same for every tuple in any given tuple. Sometimes we may not know the value of some elements in some tuples.

We use the same notation for a tuple and for a vector. Most of our data will be vectors. We write a vector in bold, so $\mathbf{x}$ could represent a vector or a tuple (the context will make it obvious which is intended).

The entire data set is $\{\mathbf{x}\}$. When we need to refer to the i'th data item, we write $\mathbf{x}_i$. Assume we have N data items, and we wish to make a new dataset out of them; we write the dataset made out of these items as $\{\mathbf{x}_i\}$ (the i is to suggest you are taking a set of items and making a dataset out of them). If we need to refer to the j'th component of a vector $\mathbf{x}_i$, we will write $x_i^{(j)}$ (notice this isn't in bold, because it is a component not a vector, and the j is in parentheses because it isn't a power). Vectors are always column vectors.

Terms:

- mean $(\{x\})$ is the mean of the dataset $\{x\}$ (definition 1, page 17).
- std (x) is the standard deviation of the dataset $\{x\}$ (definition 2, page 18).
- var $(\{x\})$ is the standard deviation of the dataset $\{x\}$ (definition 3, page 22).
- median $(\{x\})$ is the standard deviation of the dataset $\{x\}$ (definition 4, page 23).
- percentile$(\{x\}, k)$ is the $k\%$ percentile of the dataset $\{x\}$ (definition 5, page 25).
- iqr$\{x\}$ is the interquartile range of the dataset $\{x\}$ (definition 7, page 25).
- $\{\hat{x}\}$ is the dataset $\{x\}$, transformed to standard coordinates (definition 8, page 30).
- Standard normal data is defined in definition 9, (page 30).
- Normal data is defined in definition 10, (page 31).
- corr $(\{(x, y)\})$ is the correlation between two components x and y of a dataset (definition 1, page 65).
- $\emptyset$ is the empty set.
- Ω is the set of all possible outcomes of an experiment.
- Sets are written as $\mathcal{A}$.

- $\mathcal{A}^c$ is the complement of the set $\mathcal{A}$ (i.e. $\Omega - \mathcal{A}$).
- $\mathcal{E}$ is an event (page 364).
- $P(\{\mathcal{E}\})$ is the probability of event $\mathcal{E}$ (page 364).
- $P(\{\mathcal{E}\}|\{\mathcal{F}\})$ is the probability of event $\mathcal{E}$, conditioned on event $\mathcal{F}$ (page 364).
- $p(x)$ is the probability that random variable X will take the value x; also written $P(\{X = x\})$ (page 364).
- $p(x, y)$ is the probability that random variable X will take the value x and random variable Y will take the value y; also written $P(\{X = x\} \cap \{Y = y\})$ (page 364).
- $\underset{x}{\text{argmax}}\ f(x)$ means the value of x that maximises $f(x)$.
- $\underset{x}{\text{argmin}}\ f(x)$ means the value of x that minimises $f(x)$.
- $\max_i(f(x_i))$ means the largest value that f takes on the different elements of the dataset $\{x_i\}$.
- $\hat{\theta}$ is an estimated value of a parameter θ.

Background information:

- *Cards:* A standard deck of playing cards contains 52 cards. These cards are divided into four suits. The suits are: spades and clubs (which are black); and hearts and diamonds (which are red). Each suit contains 13 cards: Ace, 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack (sometimes called Knave), Queen and King. It is common to call Jack, Queen and King *court cards*.
- *Dice:* If you look hard enough, you can obtain dice with many different numbers of sides (though I've never seen a three sided die). We adopt the convention that the sides of an N sided die are labeled with the numbers $1 \ldots N$, and that no number is used twice. Most dice are like this.
- *Fairness:* Each face of a fair coin or die has the same probability of landing upmost in a flip or roll.

1.1 SOME USEFUL MATHEMATICAL FACTS

The gamma function $\Gamma(x)$ is defined by a series of steps. First, we have that for n an integer,

$$\Gamma(n) = (n-1)!$$

and then for z a complex number with positive real part (which includes positive real numbers), we have

$$\Gamma(z) = \int_0^\infty t^z \frac{e^{-t}}{t} dt.$$

By doing this, we get a function on positive real numbers that is a smooth interpolate of the factorial function. We won't do any real work with this function, so won't expand on this definition. In practice, we'll either look up a value in tables or require a software environment to produce it.

1.2 ACKNOWLEDGEMENTS

Typos spotted by: Han Chen (numerous!), Yusuf Sobh, Scott Walters, Eric Huber, — Your Name Here —

CHAPTER 2

First Tools for Looking at Data

The single most important question for a working scientist — perhaps the single most useful question anyone can ask — is: "what's going on here?" Answering this question requires creative use of different ways to make pictures of datasets, to summarize them, and to expose whatever structure might be there. This is an activity that is sometimes known as "Descriptive Statistics". There isn't any fixed recipe for understanding a dataset, but there is a rich variety of tools we can use to get insights.

2.1 DATASETS

A dataset is a collection of descriptions of different instances of the same phenomenon. These descriptions could take a variety of forms, but it is important that they are descriptions of the same thing. For example, my grandfather collected the daily rainfall in his garden for many years; we could collect the height of each person in a room; or the number of children in each family on a block; or whether 10 classmates would prefer to be "rich" or "famous". There could be more than one description recorded for each item. For example, when he recorded the contents of the rain gauge each morning, my grandfather could have recorded (say) the temperature and barometric pressure. As another example, one might record the height, weight, blood pressure and body temperature of every patient visiting a doctor's office.

The descriptions in a dataset can take a variety of forms. A description could be **categorical**, meaning that each data item can take a small set of prescribed values. For example, we might record whether each of 100 passers-by preferred to be "Rich" or "Famous". As another example, we could record whether the passers-by are "Male" or "Female". Categorical data could be **ordinal**, meaning that we can tell whether one data item is larger than another. For example, a dataset giving the number of children in a family for some set of families is categorical, because it uses only non-negative integers, but it is also ordinal, because we can tell whether one family is larger than another.

Some ordinal categorical data appears not to be numerical, but can be assigned a number in a reasonably sensible fashion. For example, many readers will recall being asked by a doctor to rate their pain on a scale of 1 to 10 — a question that is usually relatively easy to answer, but is quite strange when you think about it carefully. As another example, we could ask a set of users to rate the usability of an interface in a range from "very bad" to "very good", and then record that using -2 for "very bad", -1 for "bad", 0 for "neutral", 1 for "good", and 2 for "very good".

Many interesting datasets involve **continuous** variables (like, for example, height or weight or body temperature) when you could reasonably expect to encounter any value in a particular range. For example, we might have the heights of

all people in a particular room; or the rainfall at a particular place for each day of the year; or the number of children in each family on a list.

You should think of a dataset as a collection of d-tuples (a d-tuple is an ordered list of d elements). Tuples differ from vectors, because we can always add and subtract vectors, but we cannot necessarily add or subtract tuples. We will always write N for the number of tuples in the dataset, and d for the number of elements in each tuple. The number of elements will be the same for every tuple, though sometimes we may not know the value of some elements in some tuples (which means we must figure out how to predict their values, which we will do much later).

Index	net worth
1	100, 360
2	109, 770
3	96, 860
4	97, 860
5	108, 930
6	124, 330
7	101, 300
8	112, 710
9	106, 740
10	120, 170

Index	Taste score	Index	Taste score
1	12.3	11	34.9
2	20.9	12	57.2
3	39	13	0.7
4	47.9	14	25.9
5	5.6	15	54.9
6	25.9	16	40.9
7	37.3	17	15.9
8	21.9	18	6.4
9	18.1	19	18
10	21	20	38.9

TABLE 2.1: *On the* **left***, net worths of people you meet in a bar, in US $; I made this data up, using some information from the US Census. The index column, which tells you which data item is being referred to, is usually not displayed in a table because you can usually assume that the first line is the first item, and so on. On the* **right***, the taste score (I'm not making this up; higher is better) for 20 different cheeses. This data is real (i.e. not made up), and it comes from* `http://lib.stat.cmu.edu/DASL/Datafiles/Cheese.html`.

Each element of a tuple has its own type. Some elements might be categorical. For example, one dataset we shall see several times records entries for Gender; Grade; Age; Race; Urban/Rural; School; Goals; Grades; Sports; Looks; and Money for 478 children, so $d = 11$ and $N = 478$. In this dataset, each entry is categorical data. Clearly, these tuples are not vectors because one cannot add or subtract (say) Genders.

Most of our data will be vectors. We use the same notation for a tuple and for a vector. We write a vector in bold, so $\mathbf{x}$ could represent a vector or a tuple (the context will make it obvious which is intended).

The entire data set is $\{\mathbf{x}\}$. When we need to refer to the i'th data item, we write $\mathbf{x}_i$. Assume we have N data items, and we wish to make a new dataset out of them; we write the dataset made out of these items as $\{\mathbf{x}_i\}$ (the i is to suggest you are taking a set of items and making a dataset out of them).

In this chapter, we will work mainly with continuous data. We will see a variety of methods for plotting and summarizing 1-tuples. We can build these plots from a dataset of d-tuples by extracting the r'th element of each d-tuple.

Mostly, we will deal with continuous data. All through the book, we will see many datasets downloaded from various web sources, because people are so generous about publishing interesting datasets on the web. In the next chapter, we will look at 2-dimensional data, and we look at high dimensional data in chapter 11.

2.2 WHAT'S HAPPENING? - PLOTTING DATA

The very simplest way to present or visualize a dataset is to produce a table. Tables can be helpful, but aren't much use for large datasets, because it is difficult to get any sense of what the data means from a table. As a continuous example, table 2.1 gives a table of the net worth of a set of people you might meet in a bar (I made this data up). You can scan the table and have a rough sense of what is going on; net worths are quite close to $ 100, 000, and there aren't any very big or very small numbers. This sort of information might be useful, for example, in choosing a bar.

People would like to measure, record, and reason about an extraordinary variety of phenomena. Apparently, one can score the goodness of the flavor of cheese with a number (bigger is better); table 2.1 gives a score for each of thirty cheeses (I did not make up this data, but downloaded it from `http://lib.stat.cmu.edu/DASL/Datafiles/Cheese.html`). You should notice that a few cheeses have very high scores, and most have moderate scores. It's difficult to draw more significant conclusions from the table, though.

Gender	Goal	Gender	Goal
boy	Sports	girl	Sports
boy	Popular	girl	Grades
girl	Popular	boy	Popular
girl	Popular	boy	Popular
girl	Popular	boy	Popular
girl	Popular	girl	Grades
girl	Popular	girl	Sports
girl	Grades	girl	Popular
girl	Sports	girl	Grades
girl	Sports	girl	Sports

TABLE 2.2: *Chase and Dunner (?) collected data on what students thought made other students popular. As part of this effort, they collected information on (a) the gender and (b) the goal of students. This table gives the gender ("boy" or "girl") and the goal (to make good grades —"Grades"; to be popular — "Popular"; or to be good at sports — "Sports"). The table gives this information for the first 20 of 478 students; the rest can be found at* `http://lib.stat.cmu.edu/DASL/Datafiles/PopularKids.html`. *This data is clearly categorical, and not ordinal.*

Table 2.2 shows a table for a set of categorical data. Psychologists collected data from students in grades 4-6 in three school districts to understand what factors students thought made other students popular. This fascinating data set can be found at `http://lib.stat.cmu.edu/DASL/Datafiles/PopularKids.html`, and was prepared by Chase and Dunner (**?**). Among other things, for each student

they asked whether the student's goal was to make good grades ("Grades", for short); to be popular ("Popular"); or to be good at sports ("Sports"). They have this information for 478 students, so a table would be very hard to read. Table 2.2 shows the gender and the goal for the first 20 students in this group. It's rather harder to draw any serious conclusion from this data, because the full table would be so big. We need a more effective tool than eyeballing the table.

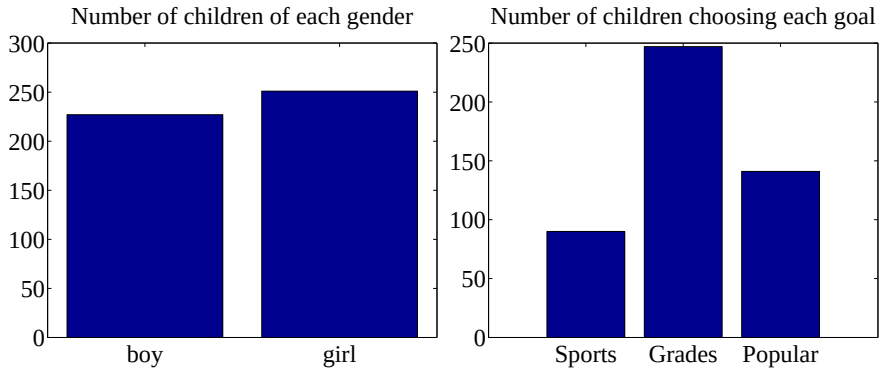

FIGURE 2.1: *On the* **left**, *a bar chart of the number of children of each gender in the Chase and Dunner study (). Notice that there are about the same number of boys and girls (the bars are about the same height). On the* **right**, *a bar chart of the number of children selecting each of three goals. You can tell, at a glance, that different goals are more or less popular by looking at the height of the bars.*

2.2.1 Bar Charts

A **bar chart** is a set of bars, one per category, where the height of each bar is proportional to the number of items in that category. A glance at a bar chart often exposes important structure in data, for example, which categories are common, and which are rare. Bar charts are particularly useful for categorical data. Figure 2.1 shows such bar charts for the genders and the goals in the student dataset of Chase and Dunner (). You can see at a glance that there are about as many boys as girls, and that there are more students who think grades are important than students who think sports or popularity is important. You couldn't draw either conclusion from Table 2.2, because I showed only the first 20 items; but a 478 item table is very difficult to read.

2.2.2 Histograms

Data is continuous when a data item could take any value in some range or set of ranges. In turn, this means that we can reasonably expect a continuous dataset contains few or no pairs of items that have *exactly* the same value. Drawing a bar chart in the obvious way — one bar per value — produces a mess of unit height bars, and seldom leads to a good plot. Instead, we would like to have fewer bars, each representing more data items. We need a procedure to decide which data items count in which bar.

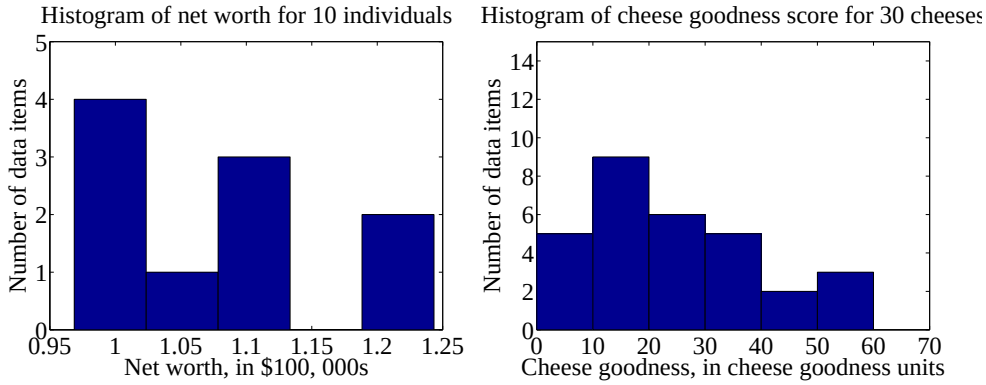

FIGURE 2.2: *On the* **left**, *a histogram of net worths from the dataset described in the text and shown in table 2.1. On the* **right**, *a histogram of cheese goodness scores from the dataset described in the text and shown in table 2.1.*

A simple generalization of a bar chart is a **histogram**. We divide the range of the data into intervals, which do not need to be equal in length. We think of each interval as having an associated pigeonhole, and choose one pigeonhole for each data item. We then build a set of boxes, one per interval. Each box sits on its interval on the horizontal axis, and its height is determined by the number of data items in the corresponding pigeonhole. In the simplest histogram, the intervals that form the bases of the boxes are equally sized. In this case, the height of the box is given by the number of data items in the box.

Figure 2.2 shows a histogram of the data in table 2.1. There are five bars — by my choice; I could have plotted ten bars — and the height of each bar gives the number of data items that fall into its interval. For example, there is one net worth in the range between $102, 500 and $107, 500. Notice that one bar is invisible, because there is no data in that range. This picture suggests conclusions consistent with the ones we had from eyeballing the table — the net worths tend to be quite similar, and around $100, 000.

Figure 2.2 shows a histogram of the data in table 2.1. There are six bars (0-10, 10-20, and so on), and the height of each bar gives the number of data items that fall into its interval — so that, for example, there are 9 cheeses in this dataset whose score is greater than or equal to 10 and less than 20. You can also use the bars to estimate other properties. So, for example, there are 14 cheeses whose score is less than 20, and 3 cheeses with a score of 50 or greater. This picture is much more helpful than the table; you can see at a glance that quite a lot of cheeses have relatively low scores, and few have high scores.

2.2.3 How to Make Histograms

Usually, one makes a histogram by finding the appropriate command or routine in your programming environment. I use Matlab, and chapter 3 sketches some useful Matlab commands. However, it is useful to understand the procedures used.

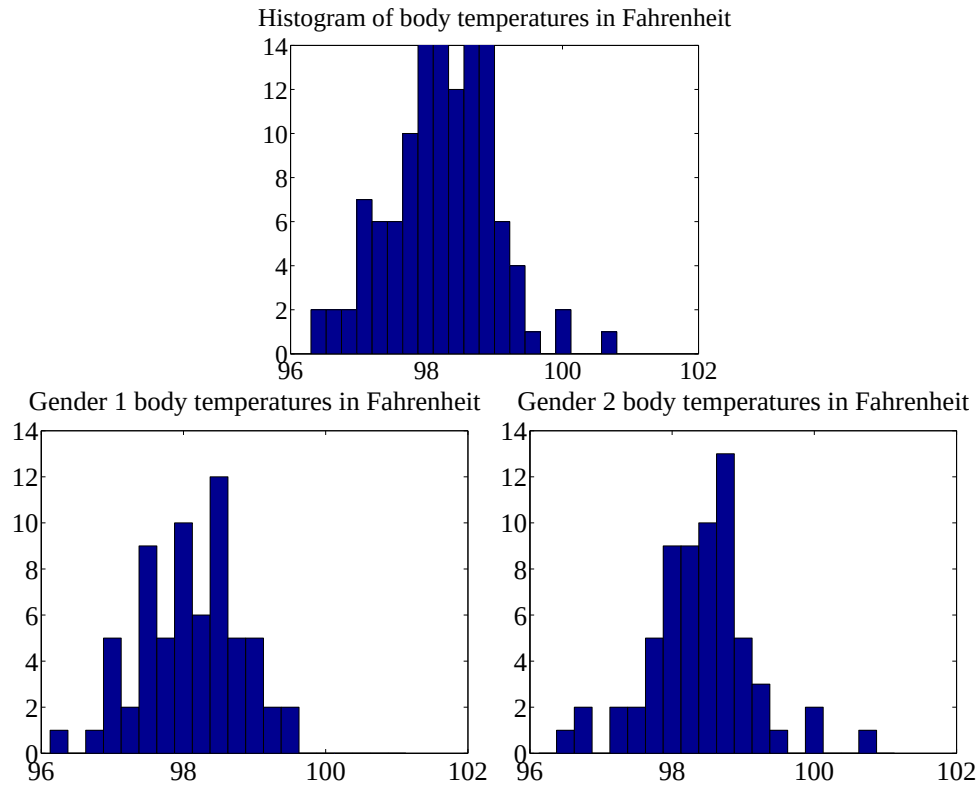

FIGURE 2.3: *On* **top**, *a histogram of body temperatures, from the dataset published at* `http://www2.stetson.edu/~jrasp/data.htm`. *These seem to be clustered fairly tightly around one value. The* **bottom row** *shows histograms for each gender (I don't know which is which). It looks as though one gender runs slightly cooler than the other.*

Histograms with Even Intervals: The easiest histogram to build uses equally sized intervals. Write x_i for the i'th number in the dataset, $x_{\min}$ for the smallest value, and $x_{\max}$ for the largest value. We divide the range between the smallest and largest values into n intervals of even width $(x_{\max} - x_{\min})/n$. In this case, the height of each box is given by the number of items in that interval. We could represent the histogram with an n-dimensional vector of counts. Each entry represents the count of the number of data items that lie in that interval. Notice we need to be careful to ensure that each point in the range of values is claimed by exactly one interval. For example, we could have intervals of $[0-1)$ and $[1-2)$, or we could have intervals of $(0-1]$ and $(1-2]$. We could *not* have intervals of $[0-1]$ and $[1-2]$, because then a data item with the value 1 would appear in two boxes. Similarly, we could not have intervals of $(0-1)$ and $(1-2)$, because then a data item with the value 1 would not appear in any box.

Histograms with Uneven Intervals: For a histogram with even intervals, it is natural that the height of each box is the number of data items in that box.

But a histogram with even intervals can have empty boxes (see figure 2.2). In this case, it can be more informative to have some larger intervals to ensure that each interval has some data items in it. But how high should we plot the box? Imagine taking two consecutive intervals in a histogram with even intervals, and fusing them. It is natural that the height of the fused box should be the average height of the two boxes. This observation gives us a rule.

Write dx for the width of the intervals; n_1 for the height of the box over the first interval (which is the number of elements in the first box); and n_2 for the height of the box over the second interval. The height of the fused box will be $(n_1 + n_2)/2$. Now the *area* of the first box is $n_1 dx$; of the second box is $n_2 dx$; and of the fused box is $(n_1 + n_2)dx$. For each of these boxes, the *area* of the box is proportional to the number of elements in the box. This gives the correct rule: plot boxes such that the area of the box is proportional to the number of elements in the box.

2.2.4 Conditional Histograms

Most people believe that normal body temperature is 98.4^o in Fahrenheit. If you take other people's temperatures often (for example, you might have children), you know that some individuals tend to run a little warmer or a little cooler than this number. I found data giving the body temperature of a set of individuals at `http://www2.stetson.edu/~jrasp/data.htm`. As you can see from the histogram (figure 2.3), the body temperatures cluster around a small set of numbers. But what causes the variation?

One possibility is gender. We can investigate this possibility by comparing a histogram of temperatures for males with histogram of temperatures for females. Such histograms are sometimes called **conditional histograms** or **class-conditional histograms**, because each histogram is conditioned on something (in this case, the histogram uses only data that comes from gender).

The dataset gives genders (as 1 or 2 - I don't know which is male and which female). Figure 2.3 gives the class conditional histograms. It does seem like individuals of one gender run a little cooler than individuals of the other, although we don't yet have mechanisms to test this possibility in detail (chapter 1).

2.3 SUMMARIZING 1D DATA

For the rest of this chapter, we will assume that data items take values that are continuous real numbers. Furthermore, we will assume that values can be added, subtracted, and multiplied by constants in a meaningful way. Human heights are one example of such data; you can add two heights, and interpret the result as a height (perhaps one person is standing on the head of the other). You can subtract one height from another, and the result is meaningful. You can multiply a height by a constant — say, 1/2 — and interpret the result (A is half as high as B). Not all data is like this. Categorical data is often not like this. For example, you could not add "Grades" to "Popular" in any useful way.

2.3.1 The Mean

One simple and effective summary of a set of data is its **mean**. This is sometimes known as the **average** of the data.

Definition: 2.1 *Mean*

Assume we have a dataset $\{x\}$ of N data items, $x_1, \ldots, x_N$. Their mean is

$$\text{mean}\left(\{x\}\right) = \frac{1}{N}\sum_{i=1}^{i=N} x_i.$$

For example, assume you're in a bar, in a group of ten people who like to talk about money. They're average people, and their net worth is given in table 2.1 (you can choose who you want to be in this story). The mean of this data is \$107, 903.

An important interpretation of the mean is that it is the best guess of the value of a new data item, given no information at all. In the bar example, if a new person walked into this bar, and you had to guess that person's net worth, you should choose \$107, 903.

Properties of the Mean The mean has several important properties you should remember:

- Scaling data scales the mean: or $\text{mean}\left(\{kx_i\}\right) = k\text{mean}\left(\{x_i\}\right)$.
- Translating data translates the mean: or $\text{mean}\left(\{x_i + c\}\right) = \text{mean}\left(\{x_i\}\right) + c$.
- The sum of signed differences from the mean is zero. This means that

$$\sum_{i=1}^{N}(x_i - \text{mean}\left(\{x_i\}\right)) = 0.$$

- Choose the number μ such that the sum of squared distances of data points to μ is minimized. That number is the mean. In notation

$$\underset{\mu}{\arg\min} \sum_{i}(x_i - \mu)^2 = \text{mean}\left(\{x_i\}\right)$$

These properties are easy to prove (and so easy to remember). All but one proof is relegated to the exercises.

Proposition: $\underset{\mu}{arg\ min}\ \sum_i (x_i - \mu)^2 = \mathsf{mean}\,(\{x\})$

Proof: Choose the number μ such that the sum of squared distances of data points to μ is minimized. That number is the mean. In notation:

$$\underset{\mu}{\arg\min}\ \sum_i (x_i - \mu)^2 = \mathsf{mean}\,(\{x\})$$

We can show this by actually minimizing the expression. We must have that the derivative of the expression we are minimizing is zero at the value of μ we are seeking. So we have

$$\begin{aligned}\frac{d}{d\mu}\sum_{i=1}^{N}(x_i - \mu)^2 &= \sum_{i=1}^{N} 2(x_i - \mu) \\ &= 2\sum_{i=1}^{N}(x_i - \mu) \\ &= 0\end{aligned}$$

so that $2N\mathsf{mean}\,(\{x\}) - 2N\mu = 0$, which means that $\mu = \mathsf{mean}\,(\{x\})$.

Property 2.1: The Average Squared Distance to the Mean is Minimized

2.3.2 Standard Deviation and Variance

We would also like to know the extent to which data items are close to the mean. This information is given by the **standard deviation**, which is the root mean square of the offsets of data from the mean.

Definition: 2.2 *Standard deviation*

Assume we have a dataset $\{x\}$ of N data items, $x_1, \ldots, x_N$. The standard deviation of this dataset is is:

$$\mathsf{std}\,(x_i) = \sqrt{\frac{1}{N}\sum_{i-1}^{i=N}(x_i - \mathsf{mean}\,(\{x\}))^2} = \sqrt{\mathsf{mean}\,(\{(x_i - \mathsf{mean}\,(\{x\}))^2\})}.$$

You should think of the standard deviation as a scale. It measures the size of the average deviation from the mean for a dataset. When the standard deviation of a dataset is large, there are many items with values much larger than, or much smaller than, the mean. When the standard deviation is small, most data items have values close to the mean. This means it is helpful to talk about how many standard devations away from the mean a particular data item is. Saying that data

item x_j is "within k standard deviations from the mean" means that

$$\mathsf{abs}\left(x_j - \mathsf{mean}\left(\{x\}\right)\right) \leq k\mathsf{std}\left(x_i\right).$$

Similarly, saying that data item x_j is "more than k standard deviations from the mean" means that

$$\mathsf{abs}\left(x_i - \mathsf{mean}\left(\{x\}\right)\right) > k\mathsf{std}\left(x\right).$$

As I will show below, there must be some data at least one standard deviation away from the mean, and there can be very few data items that are many standard deviations away from the mean.

Properties of the Standard Deviation Standard deviation has very important properties:

- Translating data does not change the standard deviation, i.e. $\mathsf{std}\left(x_i + c\right) = \mathsf{std}\left(x_i\right)$.
- Scaling data scales the standard deviation, i.e. $\mathsf{std}\left(kx_i\right) = k\mathsf{std}\left(x_i\right)$.
- For any dataset, there can be only a few items that are many standard deviations away from the mean. In particular, assume we have N data items, x_i, whose standard deviation is σ. Then there are at most $\frac{1}{k^2}$ data points lying k or more standard deviations away from the mean.
- For any dataset, there must be at least one data item that is at least one standard deviation away from the mean.

The first two properties are easy to prove, and are relegated to the exercises.

Proposition: *Assume we have a dataset $\{x\}$ of N data items, $x_1, \ldots, x_N$. Assume the standard deviation of this dataset is $\mathsf{std}(x) = \sigma$. Then there are at most $\frac{1}{k^2}$ data points lying k or more standard deviations away from the mean.*

Proof: Assume the mean is zero. There is no loss of generality here, because translating data translates the mean, but doesn't change the standard deviation. The way to prove this is to construct a dataset with the largest possible fraction r of data points lying k or more standard deviations from the mean. To achieve this, our data should have $N(1-r)$ data points each with the value 0, because these contribute 0 to the standard deviation. It should have Nr data points with the value $k\sigma$; if they are further from zero than this, each will contribute more to the standard deviation, so the fraction of such points will be fewer. Because

$$\mathsf{std}(x) = \sigma = \sqrt{\frac{\sum_i x_i^2}{N}}$$

we have that, for this rather specially constructed dataset,

$$\sigma = \sqrt{\frac{Nrk^2\sigma^2}{N}}$$

so that

$$r = \frac{1}{k^2}.$$

We constructed the dataset so that r would be as large as possible, so

$$r \leq \frac{1}{k^2}$$

for any kind of data at all.

Property 2.2: For any dataset, it is hard for data items to get many standard deviations away from the mean.

The bound of box 2.3.2 is true for *any kind of data*. This bound implies that, for example, at most 100% of *any* dataset could be one standard deviation away from the mean, 25% of *any* dataset is 2 standard deviations away from the mean and at most 11% of *any* dataset could be 3 standard deviations away from the mean. But the configuration of data that achieves this bound is very unusual. This means the bound tends to wildly overstate how much data is far from the mean for most practical datasets. Most data has more random structure, meaning that we expect to see very much *less* data far from the mean than the bound predicts. For example, much data can reasonably be modelled as coming from a normal distribution (a topic we'll go into later). For such data, we expect that about 68% of the data is within one standard deviation of the mean, 95% is within two standard deviations of the mean, and 99.7% is within three standard deviations of the mean, and the percentage of data that is within ten standard deviations of the mean is essentially indistinguishable from 100%. This kind of behavior is quite common; the crucial point about the standard deviation is that you won't see much

data that lies many standard deviations from the mean, because you can't.

Proposition:

$$(\mathsf{std}\,(x))^2 \leq \max_i (x_i - \mathsf{mean}\,(\{x\}))^2.$$

Proof: You can see this by looking at the expression for standard deviation. We have

$$\mathsf{std}\,(x) = \sqrt{\frac{1}{N}\sum_{i-1}^{i=N}(x_i - \mathsf{mean}\,(\{x\}))^2}.$$

Now, this means that

$$N(\mathsf{std}\,(x))^2 = \sum_{i-1}^{i=N}(x_i - \mathsf{mean}\,(\{x\}))^2.$$

But

$$\sum_{i-1}^{i=N}(x_i - \mathsf{mean}\,(\{x\}))^2 \leq N \max_i (x_i - \mathsf{mean}\,(\{x\}))^2$$

so

$$(\mathsf{std}\,(x))^2 \leq \max_i (x_i - \mathsf{mean}\,(\{x\}))^2.$$

Property 2.3: For any dataset, there must be at least one data item that is at least one standard deviation away from the mean.

Boxes 2.3.2 and 2.3.2 mean that the standard deviation is quite informative. Very little data is many standard deviations away from the mean; similarly, at least some of the data should be one or more standard deviations away from the mean. So the standard deviation tells us how data points are scattered about the mean.

Potential point of confusion: *There is an ambiguity that comes up often here because two (very slightly) different numbers are called the standard deviation of a dataset.*

What is going on here: One — the one we use in this chapter — is an estimate of the scale of the data, as we describe it. The other differs from our expression very slightly; one computes

$$\sqrt{\frac{\sum_i (x_i - \text{mean}\left(\{x\}\right))^2}{N-1}}$$

(notice the $N-1$ for our N). If N is large, this number is basically the same as the number we compute, but for smaller N there is a difference that can be significant. Irritatingly, this number is also called the standard deviation; even more irritatingly, we will have to deal with it, but not yet. I mention it now because you may look up terms I have used, find this definition, and wonder whether I know what I'm talking about. The confusion arises because sometimes the datasets we see are actually samples of larger datasets. For example, in some circumstances you could think of the net worth dataset as a sample of all the net worths in the USA. In such cases, we are often interested in the standard deviation *of the dataset that was sampled*. The second number is a slightly better way to estimate this standard deviation than the definition we have been working with. Don't worry - the N in our expressions is the right thing to use for what we're doing.

2.3.3 Variance

It turns out that thinking in terms of the square of the standard deviation, which is known as the **variance**, will allow us to generalize our summaries to apply to higher dimensional data.

Definition: 2.3 *Variance*

Assume we have a dataset $\{x\}$ of N data items, $x_1, \ldots, x_N$. where $N > 1$. Their variance is:

$$\text{var}\left(\{x\}\right) = \frac{1}{N}\left(\sum_{i-1}^{i=N}(x_i - \text{mean}\left(\{x\}\right))^2\right) = \text{mean}\left(\left\{(x_i - \text{mean}\left(\{x\}\right))^2\right\}\right).$$

One good way to think of the variance is as the mean-square error you would

incur if you replaced each data item with the mean. Another is that it is the square of the standard deviation.

Properties of the Variance The properties of the variance follow from the fact that it is the square of the standard deviation. We have that:

- Translating data does not change the variance, i.e. $\mathsf{var}\left(\{x + c\}\right) = \mathsf{var}\left(\{x\}\right)$.
- Scaling data scales the variance by a square of the scale, i.e. $\mathsf{var}\left(\{kx\}\right) = k^2\mathsf{var}\left(\{x\}\right)$.

While one could restate the other two properties of the standard deviation in terms of the variance, it isn't really natural to do so. The standard deviation is in the same units as the original data, and should be thought of as a scale. Because the variance is the square of the standard deviation, it isn't a natural scale (unless you take its square root!).

2.3.4 The Median

One problem with the mean is that it can be affected strongly by extreme values. Go back to the bar example, of section 2.3.1. Now Warren Buffett (or Bill Gates, or your favorite billionaire) walks in. What happened to the average net worth?

Assume your billionaire has net worth \$ 1, 000, 000, 000. Then the mean net worth suddenly has become

$$\frac{10 \times \$107,903 + \$1,000,000,000}{11} = \$91,007,184$$

But this mean isn't a very helpful summary of the people in the bar. It is probably more useful to think of the net worth data as ten people together with one billionaire. The billionaire is known as an **outlier**.

One way to get outliers is that a small number of data items are very different, due to minor effects you don't want to model. Another is that the data was misrecorded, or mistranscribed. Another possibility is that there is just too much variation in the data to summarize it well. For example, a small number of extremely wealthy people could change the average net worth of US residents dramatically, as the example shows. An alternative to using a mean is to use a **median**.

Definition: 2.4 *Median*

The median of a set of data points is obtained by sorting the data points, and finding the point halfway along the list. If the list is of even length, it's usual to average the two numbers on either side of the middle. We write

$$\mathsf{median}\left(\{x_i\}\right)$$

for the operator that returns the median.

For example,

$$\mathsf{median}\left(\{3,5,7\}\right) = 5,$$

$$\mathsf{median}\left(\{3,4,5,6,7\}\right) = 5,$$

and

$$\mathsf{median}\left(\{3,4,5,6\}\right) = 4.5.$$

For much, but not all, data, you can expect that roughly half the data is smaller than the median, and roughly half is larger than the median. Sometimes this property fails. For example,

$$\mathsf{median}\left(\{1,2,2,2,2,2,2,2,3\}\right) = 2.$$

With this definition, the median of our list of net worths is $\$107,835$. If we insert the billionaire, the median becomes $\$108,930$. Notice by how little the number has changed — it remains an effective summary of the data.

Properties of the median You can think of the median of a dataset as giving the "middle" or "center" value. This means it is rather like the mean, which also gives a (slightly differently defined) "middle" or "center" value. The mean has the important properties that if you translate the dataset, the mean translates, and if you scale the dataset, the mean scales. The median has these properties, too:

- Translating data translates the median, i.e. $\mathsf{median}\left(\{x+c\}\right) = \mathsf{median}\left(\{x\}\right) + c$.
- Scaling data scales the median by the same scale, i.e. $\mathsf{median}\left(\{kx\}\right) = k\mathsf{median}\left(\{x\}\right)$.

Each is easily proved, and proofs are relegated to the exercises.

2.3.5 Interquartile Range

Outliers are a nuisance in all sorts of ways. Plotting the histogram of the net worth data with the billionaire included will be tricky. Either you leave the billionaire out of the plot, or all the histogram bars are tiny. Visualizing this plot shows outliers can affect standard deviations severely, too. For our net worth data, the standard deviation without the billionaire is $\$9265$, but if we put the billionaire in there, it is $\$3.014 \times 10^8$. When the billionaire is in the dataset, the mean is about $91M\$$ and the standard deviation is about $300M\$$ — so all but one of the data items lie about a third of a standard deviation away from the mean on the small side. The other data item (the billionaire) is about three standard deviations away from the mean on the large side. In this case, the standard deviation has done its work of informing us that there are huge changes in the data, but isn't really helpful as a description of the data.

The problem is this: describing the net worth data with billionaire as a having a mean of $\$9.101 \times 10^7$ with a standard deviation of $\$3.014 \times 10^8$ really isn't terribly helpful. Instead, the data really should be seen as a clump of values that are near $\$100,000$ and moderately close to one another, and one massive number (the billionaire outlier).

One thing we could do is simply remove the billionaire and compute mean and standard deviation. This isn't always easy to do, because it's often less obvious which points are outliers. An alternative is to follow the strategy we did when we used the median. Find a summary that describes scale, but is less affected by outliers than the standard deviation. This is the **interquartile range**; to define it, we need to define percentiles and quartiles, which are useful anyway.

Definition: 2.5 *Percentile*

The k'th percentile is the value such that $k\%$ of the data is less than or equal to that value. We write $\mathsf{percentile}(\{x\}, k)$ for the k'th percentile of dataset $\{x\}$.

Definition: 2.6 *Quartiles*

The first quartile of the data is the value such that 25% of the data is less than or equal to that value (i.e. $\mathsf{percentile}(\{x\}, 25)$). The second quartile of the data is the value such that 50% of the data is less than or equal to that value, which is usually the median (i.e. $\mathsf{percentile}(\{x\}, 50)$). The third quartile of the data is the value such that 75% of the data is less than or equal to that value (i.e. $\mathsf{percentile}(\{x\}, 75)$).

Definition: 2.7 *Interquartile Range*

The interquartile range of a dataset $\{x\}$ is $\mathsf{iqr}\{x\} = \mathsf{percentile}(\{x\}, 75) - \mathsf{percentile}(\{x\}, 25)$.

Like the standard deviation, the interquartile range gives an estimate of how widely the data is spread out. But it is quite well-behaved in the presence of outliers. For our net worth data without the billionaire, the interquartile range is \$12350; with the billionaire, it is \$17710.

Properties of the interquartile range You can think of the interquartile range of a dataset as giving an estimate of the scale of the difference from the mean. This means it is rather like the standard deviation, which also gives a (slightly differently defined) scale. The standard deviation has the important properties that if you translate the dataset, the standard deviation translates, and if you scale the dataset, the standard deviation scales. The interquartile range has these properties, too:

- Translating data does not change the interquartile range, i.e. $\mathsf{iqr}\{x + c\} =$

$\mathsf{iqr}\{x\}$.

- Scaling data scales the interquartile range by the same scale, i.e. $\mathsf{iqr}\{kx\} = k^2\mathsf{iqr}\{x\}$.

Each is easily proved, and proofs are relegated to the exercises.

2.3.6 Using Summaries Sensibly

One should be careful how one summarizes data. For example, the statement that "the average US family has 2.6 children" invites mockery (the example is from Andrew Vickers' book *What is a p-value anyway?*), because you can't have fractions of a child — no family has 2.6 children. A more accurate way to say things might be "the average of the number of children in a US family is 2.6", but this is clumsy. What is going wrong here is the 2.6 is a mean, but the number of children in a family is a categorical variable. Reporting the mean of a categorical variable is often a bad idea, because you may never encounter this value (the 2.6 children). For a categorical variable, giving the median value and perhaps the interquartile range often makes much more sense than reporting the mean.

For continuous variables, reporting the mean is reasonable because you could expect to encounter a data item with this value, even if you haven't seen one in the particular data set you have. It is sensible to look at both mean and median; if they're significantly different, then there is probably something going on that is worth understanding. You'd want to plot the data using the methods of the next section before you decided what to report.

You should also be careful about how precisely numbers are reported (equivalently, the number of significant figures). Numerical and statistical software will produce very large numbers of digits freely, but not all are always useful. This is a particular nuisance in the case of the mean, because you might add many numbers, then divide by a large number; in this case, you will get many digits, but some might not be meaningful. For example, Vickers (ibid) describes a paper reporting the mean length of pregnancy as 32.833 weeks. That fifth digit suggests we know the mean length of pregnancy to about 0.001 weeks, or roughly 10 minutes. Neither medical interviewing nor people's memory for past events is that detailed. Furthermore, when you interview them about embarrassing topics, people quite often lie. There is no prospect of knowing this number with this precision.

People regularly report silly numbers of digits because it is easy to miss the harm caused by doing so. But the harm is there: you are implying to other people, and to yourself, that you know something more accurately than you do. At some point, someone will suffer for it.

2.4 PLOTS AND SUMMARIES

Knowing the mean, standard deviation, median and interquartile range of a dataset gives us some information about what its histogram might look like. In fact, the summaries give us a language in which to describe a variety of characteristic properties of histograms that are worth knowing about (Section 2.4.1). Quite remarkably, many different datasets have the same shape of histogram (Section 2.4.2). For such data, we know roughly what percentage of data items are how far from the mean.

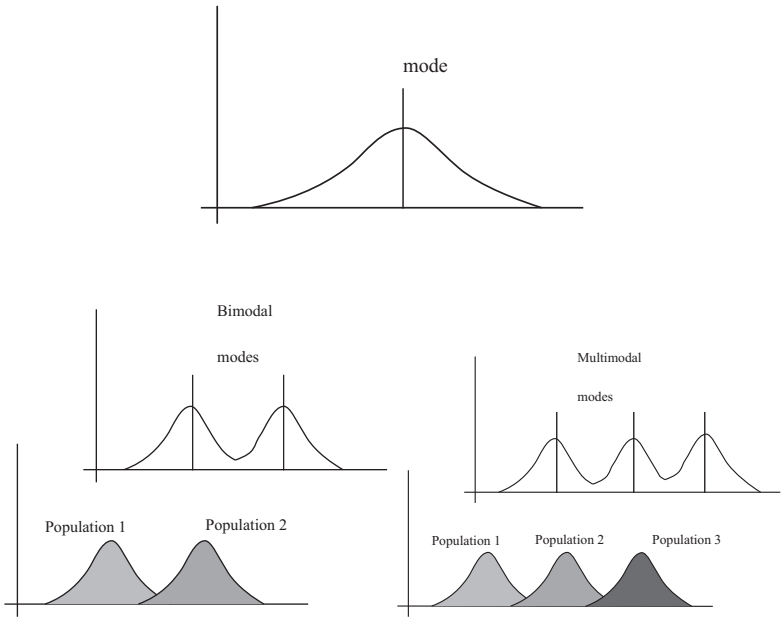

FIGURE 2.4: *Many histograms are unimodal, like the example on the* **top***; there is one peak, or mode. Some are bimodal (two peaks;* **bottom left***) or even multimodal (two or more peaks;* **bottom right***). One common reason (but not the only reason) is that there are actually two populations being conflated in the histograms. For example, measuring adult heights might result in a bimodal histogram, if male and female heights were slightly different. As another example, measuring the weight of dogs might result in a multimodal histogram if you did not distinguish between breeds (eg chihauhau, terrier, german shepherd, pyranean mountain dog, etc.).*

Complex datasets can be difficult to interpret with histograms alone, because it is hard to compare many histograms by eye. Section 2.4.3 describes a clever plot of various summaries of datasets that makes it easier to compare many cases.

2.4.1 Some Properties of Histograms

The **tails** of a histogram are the relatively uncommon values that are significantly larger (resp. smaller) than the value at the peak (which is sometimes called the **mode**). A histogram is **unimodal** if there is only one peak; if there are more than one, it is **multimodal**, with the special term **bimodal** sometimes being used for the case where there are two peaks (Figure 2.4). The histograms we have seen have been relatively symmetric, where the left and right tails are about as long as one another. Another way to think about this is that values a lot larger than the mean are about as common as values a lot smaller than the mean. Not all data is symmetric. In some datasets, one or another tail is longer (figure 2.5). This effect is called **skew**.

Skew appears often in real data. SOCR (the Statistics Online Computa-

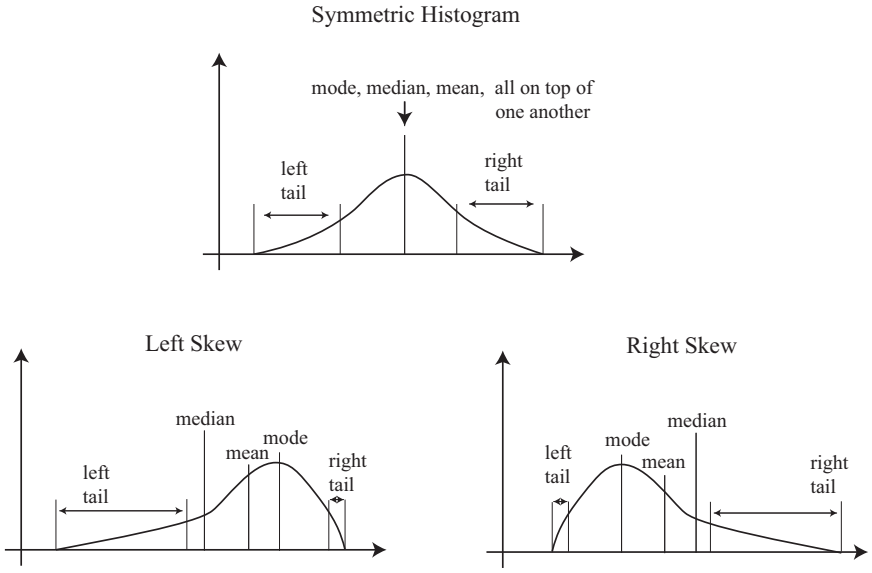

FIGURE 2.5: *On the* **top**, *an example of a symmetric histogram, showing its tails (relatively uncommon values that are significantly larger or smaller than the peak or mode).* **Lower left**, *a sketch of a left-skewed histogram. Here there are few large values, but some very small values that occur with significant frequency. We say the left tail is "long", and that the histogram is left skewed. You may find this confusing, because the main bump is to the right — one way to remember this is that the left tail has been stretched.* **Lower right**, *a sketch of a right-skewed histogram. Here there are few small values, but some very large values that occur with significant frequency. We say the right tail is "long", and that the histogram is right skewed.*

tional Resource) publishes a number of datasets. Here we discuss a dataset of citations to faculty publications. For each of five UCLA faculty members, SOCR collected the number of times each of the papers they had authored had been cited by other authors (data at `http://wiki.stat.ucla.edu/socr/index.php/SOCR_Data_Dinov_072108_H_Index_Pubs`). Generally, a small number of papers get many citations, and many papers get few citations. We see this pattern in the histograms of citation numbers (figure 2.6). These are very different from (say) the body temperature pictures. In the citation histograms, there are many data items that have very few citations, and few that have many citations. This means that the right tail of the histogram is longer, so the histogram is skewed to the right.

One way to check for skewness is to look at the histogram; another is to compare mean and median (though this is not foolproof). For the first citation histogram, the mean is 24.7 and the median is 7.5; for the second, the mean is 24.4, and the median is 11. In each case, the mean is a lot bigger than the median. Recall the definition of the median (form a ranked list of the data points, and find the point halfway along the list). For much data, the result is larger than about half

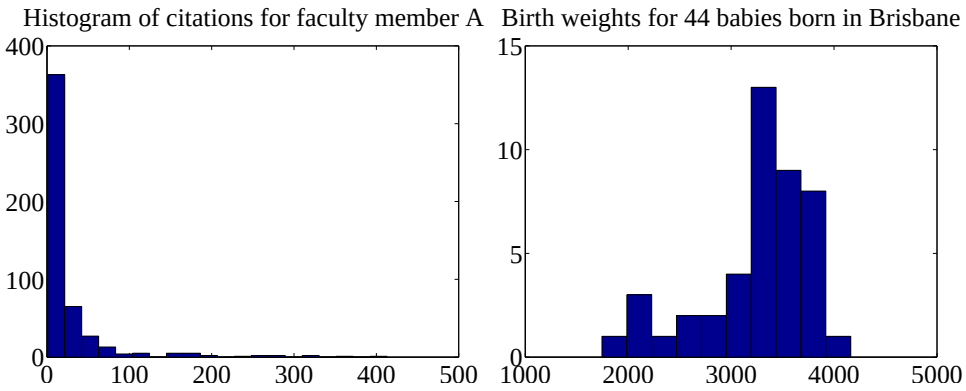

FIGURE 2.6: *On the* **left**, *a histogram of citations for a faculty member, from data at* `http://wiki.stat.ucla.edu/socr/index.php/SOCR_Data_Dinov_072108_H_Index_Pubs`. *Very few publications have many citations, and many publications have few. This means the histogram is strongly right-skewed. On the* **right**, *a histogram of birth weights for 44 babies borne in Brisbane in 1997. This histogram looks slightly left-skewed.*

of the data set and smaller than about half the dataset. So if the median is quite small compared to the mean, then there are many small data items and a small number of data items that are large — the right tail is longer, so the histogram is skewed to the right.

Left-skewed data also occurs; figure 2.6 shows a histogram of the birth weights of 44 babies born in Brisbane, in 1997 (from `http://www.amstat.org/publications/jse/jse_data_archive.htm`). This data appears to be somewhat left-skewed, as birth weights can be a lot smaller than the mean, but tend not to be much larger than the mean.

Skewed data is often, but not always, the result of constraints. For example, good obstetrical practice tries to ensure that very large birth weights are rare (birth is typically induced before the baby gets too heavy), but it may be quite hard to avoid some small birth weights. This could could skew birth weights to the left (because large babies will get born, but will not be as heavy as they could be if obstetricians had not interfered). Similarly, income data can be skewed to the right by the fact that income is always positive. Test mark data is often skewed — whether to right or left depends on the circumstances — by the fact that there is a largest possible mark and a smallest possible mark.

2.4.2 Standard Coordinates and Normal Data

It is useful to look at lots of histograms, because it is often possible to get some useful insights about data. However, in their current form, histograms are hard to compare. This is because each is in a different set of units. A histogram for length data will consist of boxes whose horizontal units are, say, metres; a histogram for mass data will consist of boxes whose horizontal units are in, say, kilograms. Furthermore, these histograms typically span different ranges.

We can make histograms comparable by (a) estimating the "location" of the plot on the horizontal axis and (b) estimating the "scale" of the plot. The location is given by the mean, and the scale by the standard deviation. We could then normalize the data by subtracting the location (mean) and dividing by the standard deviation (scale). The resulting values are unitless, and have zero mean. They are often known as **standard coordinates**.

Definition: 2.8 *Standard coordinates*

Assume we have a dataset $\{x\}$ of N data items, $x_1, \ldots, x_N$. We represent these data items in standard coordinates by computing

$$\hat{x}_i = \frac{(x_i - \mathsf{mean}\,(\{x\}))}{\mathsf{std}\,(x)}.$$

We write $\{\hat{x}\}$ for a dataset that happens to be in standard coordinates.

Standard coordinates have some important properties. Assume we have N data items. Write x_i for the i'th data item, and $\hat{x}_i$ for the i'th data item in standard coordinates (I sometimes refer to these as "normalized data items"). Then we have

$$\mathsf{mean}\,(\{\hat{x}\}) = 0.$$

We also have that

$$\mathsf{std}\,(\hat{x}) = 1.$$

An extremely important fact about data is that, for many kinds of data, histograms of these standard coordinates look the same. Many completely different datasets produce a histogram that, in standard coordinates, has a very specific appearance. It is symmetric, unimodal; it looks like a narrow bump. If there were enough data points and the histogram boxes were small enough, the curve would look like the curve in figure 2.7. This phenomenon is so important that data of this form has a special name.

Definition: 2.9 *Standard normal data*

Data is **standard normal data** if, when we have a great deal of data, the histogram is a close approximation to the **standard normal curve**. This curve is given by

$$y(x) = \frac{1}{\sqrt{2pi}} e^{\left(-x^2/2\right)}$$

(which is shown in figure 2.7).

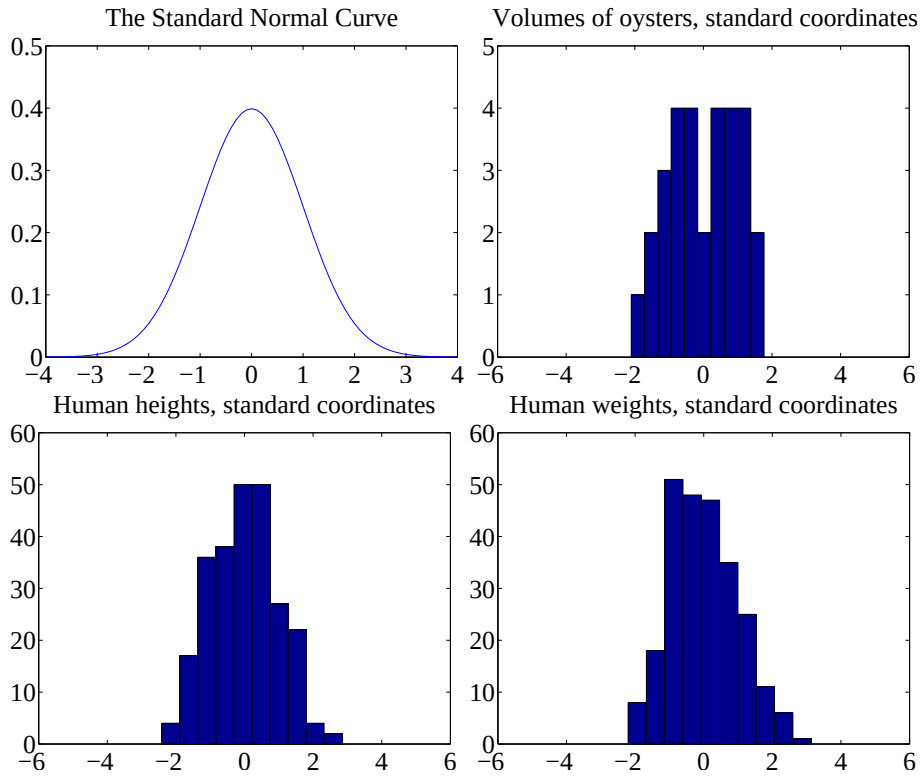

FIGURE 2.7: *Data is standard normal data when its histogram takes a stylized, bell-shaped form, plotted above. One usually requires a lot of data and very small histogram boxes for this form to be reproduced closely. Nonetheless, the histogram for normal data is unimodal (has a single bump) and is symmetric; the tails fall off fairly fast, and there are few data items that are many standard deviations from the mean. Many quite different data sets have histograms that are similar to the normal curve; I show three such datasets here.*

Definition: 2.10 *Normal data*

Data is **normal data** if, when we subtract the mean and divide by the standard deviation (i.e. compute standard coordinates), it becomes standard normal data.

It is not always easy to tell whether data is normal or not, and there are a variety of tests one can use, which we discuss later. However, there are many examples of normal data. Figure 2.7 shows a diverse variety of data sets, plotted as histograms in standard coordinates. These include: the volumes of 30 oysters (from `http://www.amstat.org/publications/jse/jse_data_archive.htm`; look

for 30oysters.dat.txt); human heights (from `http://www2.stetson.edu/~jrasp/data.htm`; look for bodyfat.xls, with two outliers removed); and human weights (from `http://www2.stetson.edu/~jrasp/data.htm`; look for bodyfat.xls, with two outliers removed).

Properties of normal data For the moment, assume we know that a dataset is normal. Then we expect it to have the following properties:

- If we normalize it, its histogram will be close to the standard normal curve. This means, among other things, that the data is not significantly skewed.
- About 68% of the data lie within one standard deviation of the mean. We will prove this later.
- About 95% of the data lie within two standard deviations of the mean. We will prove this later.
- About 99% of the data lie within three standard deviations of the mean. We will prove this later.

In turn, these properties imply that data that contains outliers (points many standard deviations away from the mean) is not normal. This is usually a very safe assumption. It is quite common to model a dataset by excluding a small number of outliers, then modelling the remaining data as normal. For example, if I exclude two outliers from the height and weight data from `http://www2.stetson.edu/~jrasp/data.htm`, the data looks pretty close to normal.

2.4.3 Boxplots

It is usually hard to compare multiple histograms by eye. One problem with comparing histograms is the amount of space they take up on a plot, because each histogram involves multiple vertical bars. This means it is hard to plot multiple overlapping histograms cleanly. If you plot each one on a separate figure, you have to handle a large number of separate figures; either you print them too small to see enough detail, or you have to keep flipping over pages.

A **boxplot** is a way to plot data that simplifies comparison. A boxplot displays a dataset as a vertical picture. There is a vertical box whose height corresponds to the interquartile range of the data (the width is just to make the figure easy to interpret). Then there is a horizontal line for the median; and the behavior of the rest of the data is indicated with whiskers and/or outlier markers. This means that each dataset makes is represented by a vertical structure, making it easy to show multiple datasets on one plot *and interpret the plot* (Figure 2.8).

To build a boxplot, we first plot a box that runs from the first to the third quartile. We then show the median with a horizontal line. We then decide which data items should be outliers. A variety of rules are possible; for the plots I show, I used the rule that data items that are larger than $q_3 + 1.5(q_3 - q_1)$ or smaller than $q_1 - 1.5(q_3 - q_1)$, are outliers. This criterion looks for data items that are more than one and a half interquartile ranges above the third quartile, or more than one and a half interquartile ranges below the first quartile.

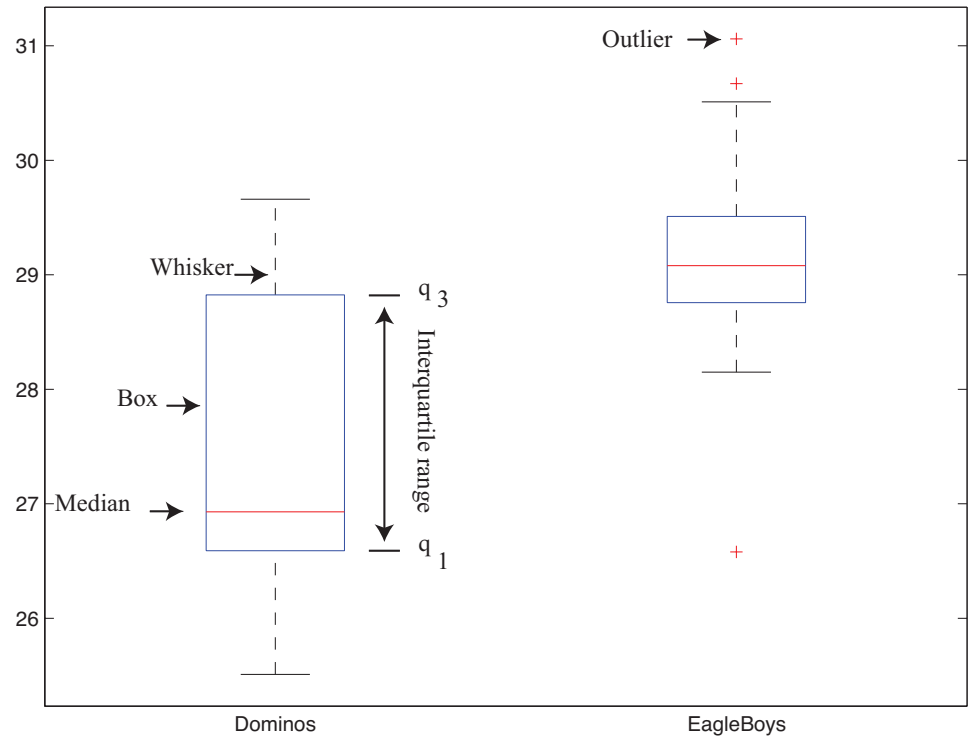

FIGURE 2.8: *A boxplot showing the box, the median, the whiskers and two outliers. Notice that we can compare the two datasets rather easily; the next section explains the comparison.*

Once we have identified outliers, we plot these with a special symbol (crosses in the plots I show). We then plot whiskers, which show the range of non-outlier data. We draw a whisker from q_1 to the smallest data item that is not an outlier, and from q_3 to the largest data item that is not an outlier. While all this sounds complicated, any reasonable programming environment will have a function that will do it for you. Figure 2.8 shows an example boxplot. Notice that the rich graphical structure means it is quite straightforward to compare two histograms.

2.5 WHOSE IS BIGGER? INVESTIGATING AUSTRALIAN PIZZAS

At `http://www.amstat.org/publications/jse/jse_data_archive.htm`), you will find a dataset giving the diameter of pizzas, measured in Australia (search for the word "pizza"). This website also gives the backstory for this dataset. Apparently, EagleBoys pizza claims that their pizzas are always bigger than Dominos pizzas, and published a set of measurements to support this claim (the measurements were available at `http://www.eagleboys.com.au/realsizepizza` as of Feb 2012, but seem not to be there anymore).

Whose pizzas are bigger? and why? A histogram of all the pizza sizes appears

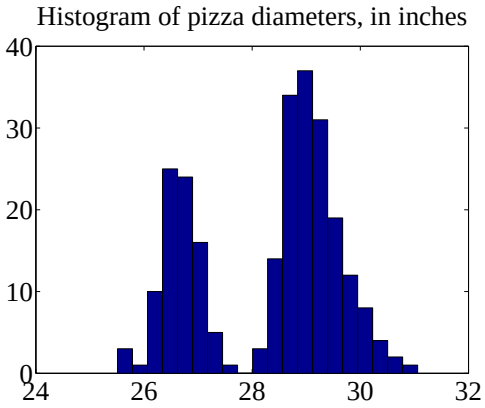

FIGURE 2.9: *A histogram of pizza diameters from the dataset described in the text. Notice that there seem to be two populations.*

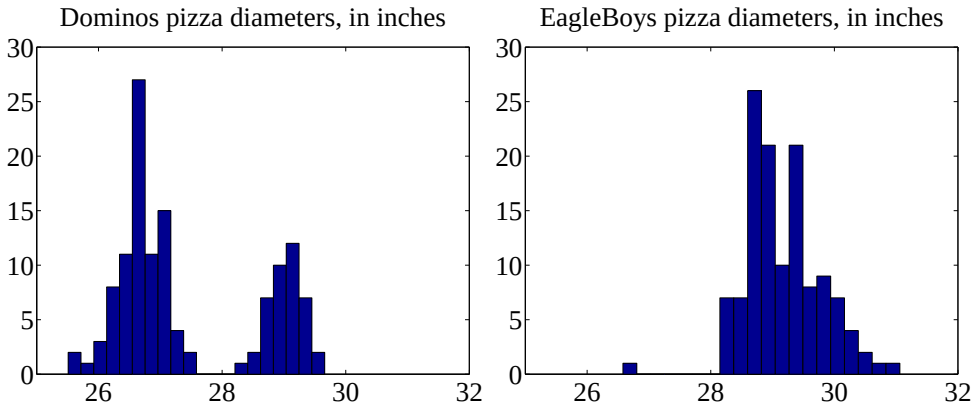

FIGURE 2.10: *On the* **left**, *the class-conditional histogram of Dominos pizza diameters from the pizza data set; on the* **right**, *the class-conditional histogram of EagleBoys pizza diameters. Notice that EagleBoys pizzas seem to follow the pattern we expect — the diameters are clustered tightly around a mean, and there is a small standard deviation — but Dominos pizzas do not seem to be like that. There is more to understand about this data.*

in figure 2.9. We would not expect every pizza produced by a restaurant to have exactly the same diameter, but the diameters are probably pretty close to one another, and pretty close to some standard value. This would suggest that we'd expect to see a histogram which looks like a single, rather narrow, bump about a mean. This is not what we see in figure 2.9 — instead, there are two bumps, which suggests two populations of pizzas. This isn't particularly surprising, because we know that some pizzas come from EagleBoys and some from Dominos.

If you look more closely at the data in the dataset, you will notice that each data item is tagged with the company it comes from. We can now easily plot

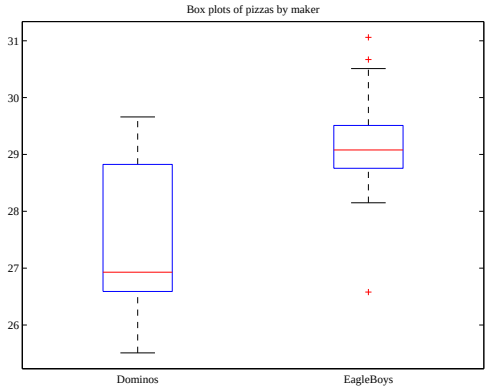

FIGURE 2.11: *Boxplots of the pizza data, comparing EagleBoys and Dominos pizza. There are several curiosities here: why is the range for Dominos so large (25.5-29)? EagleBoys has a smaller range, but has several substantial outliers; why? One would expect pizza manufacturers to try and control diameter fairly closely, because pizzas that are too small present risks (annoying customers; publicity; hostile advertising) and pizzas that are too large should affect profits.*

conditional histograms, conditioning on the company that the pizza came from. These appear in figure 2.10. Notice that EagleBoys pizzas seem to follow the pattern we expect — the diameters are clustered tightly around one value — but Dominos pizzas do not seem to be like that. This is reflected in a boxplot (figure 2.11), which shows the range of Dominos pizza sizes is surprisingly large, and that EagleBoys pizza sizes have several large outliers. There is more to understand about this data. The dataset contains labels for the type of crust and the type of topping — perhaps these properties affect the size of the pizza?

EagleBoys produces DeepPan, MidCrust and ThinCrust pizzas, and Dominos produces DeepPan, ClassicCrust and ThinNCrispy pizzas. This may have something to do with the observed patterns, but comparing six histograms by eye is unattractive. A boxplot is the right way to compare these cases (figure 2.12). The boxplot gives some more insight into the data. Dominos thin crust appear to have a narrow range of diameters (with several outliers), where the median pizza is rather larger than either the deep pan or the classic crust pizza. EagleBoys pizzas all have a range of diameters that is (a) rather similar across the types and (b) rather a lot like the Dominos thin crust. There are outliers, but few for each type.

Another possibility is that the variation in size is explained by the topping. We can compare types and toppings by producing a set of conditional boxplots (i.e. the diameters for each type and each topping). This leads to rather a lot of boxes (figure 2.13), but they're still easy to compare by eye. The main difficulty is that the labels on the plot have to be shortened. I made labels using the first letter from the manufacturer ("D" or "E"); the first letter from the crust type (previous paragraph); and the first and last letter of the topping. Toppings for Dominos are: Hawaiian; Supreme; BBQMeatlovers. For EagleBoys, toppings are: Hawaiian; SuperSupremo; and BBQMeatlovers. This gives the labels: 'DCBs'; (Dominos; Clas-

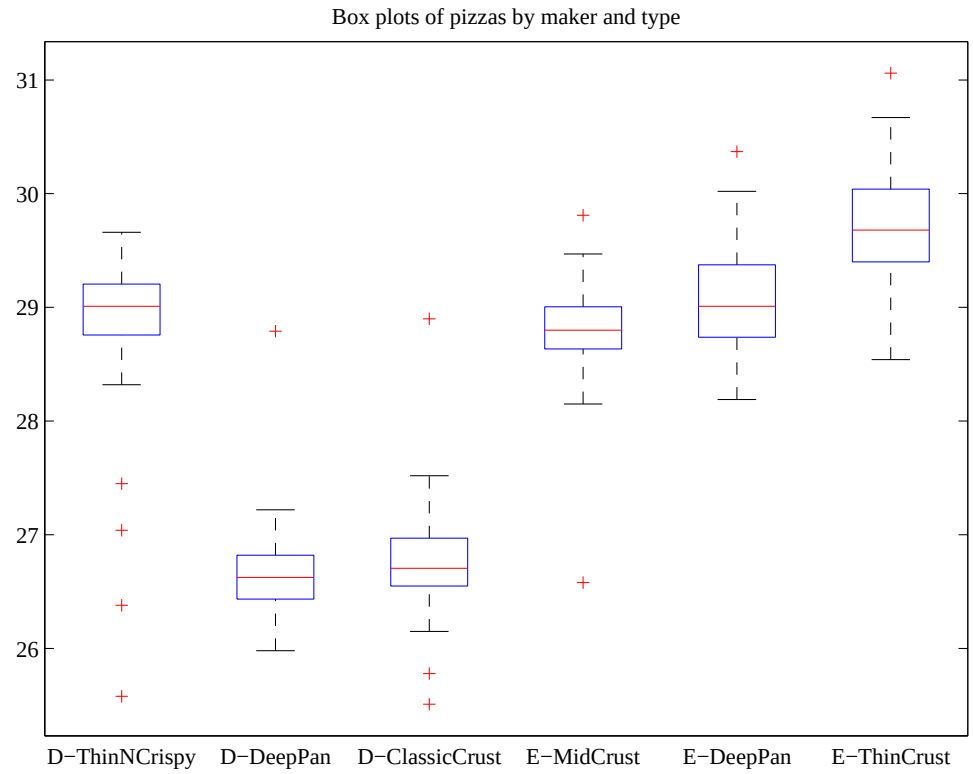

FIGURE 2.12: *Boxplots for the pizza data, broken out by type (thin crust, etc.).*

sicCrust; BBQMeatlovers); 'DCHn'; 'DCSe'; 'DDBs'; 'DDHn'; 'DDSe'; 'DTBs'; 'DTHn'; 'DTSe'; 'EDBs'; 'EDHn'; 'EDSo'; 'EMBs'; 'EMHn'; 'EMSo'; 'ETBs'; 'ETHn'; 'ETSo'. Figure 2.13 suggests that the topping isn't what is important, but the crust (group the boxplots by eye).

What could be going on here? One possible explanation is that Eagleboys have tighter control over the size of the final pizza. One way this could happen is that all EagleBoys pizzas start the same size and shrink the same amount in baking, whereas all Dominos pizzas start a standard diameter, but different Dominos crusts shrink differently in baking. Another way is that Dominos makes different size crusts for different types, but that the cooks sometimes get confused. Yet another possibility is that Dominos controls portions by the mass of dough (so thin crust diameters tend to be larger), but Eagleboys controls by the diameter of the crust.

You should notice that this is more than just a fun story. If you were a manager at a pizza firm, you'd need to make choices about how to control costs. Labor costs, rent, and portion control (i.e. how much pizza, topping, etc. a customer gets for their money) are the main thing to worry about. If the same kind of pizza has a wide range of diameters, you have a problem, because some customers are getting too much (which affects your profit) or too little (which means they might call

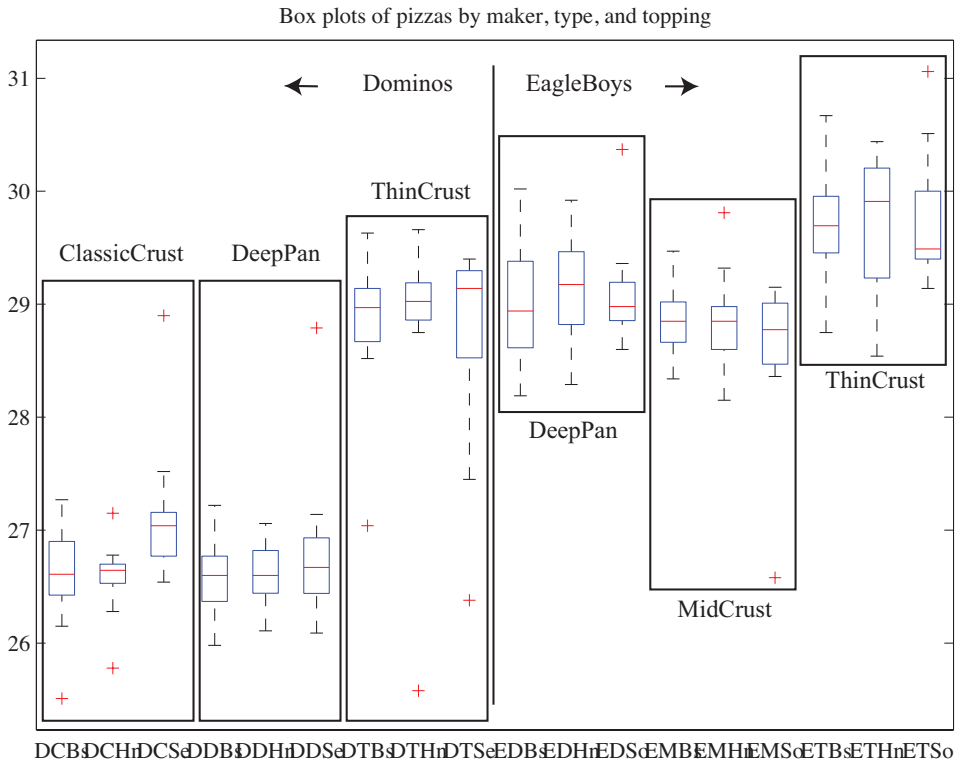

FIGURE 2.13: *The pizzas are now broken up by topping as well as crust type (look at the source for the meaning of the names). I have separated Dominos from Eagleboys with a vertical line, and grouped each crust type with a box. It looks as though the issue is not the type of topping, but the crust. Eagleboys seems to have tighter control over the size of the final pizza.*

someone else). But making more regular pizzas might require more skilled (and so more expensive) labor. The fact that Dominos and EagleBoys seem to be following different strategies successfully suggests that more than one strategy might work. But you can't choose if you don't know what's happening. As I said at the start, "what's going on here?" is perhaps the single most useful question anyone can ask.

2.6 WHAT YOU MUST REMEMBER

You should be able to:

- Plot a bar chart for a dataset.
- Plot a histogram for a dataset.
- Tell whether the histogram is skewed or not, and in which direction.
- Plot a box plot for one or several datasets.

- Interpret a box plot.

You should remember:

- The definition and properties of the mean.
- The definition and properties of the standard deviation.
- The definition and properties of the variance.
- The definition and properties of the median.
- The definition and properties of the interquartile range.

PROBLEMS

2.1. Show that $\mathsf{mean}\,(\{kx\}) = k\mathsf{mean}\,(\{x\})$ by substituting into the definition.
2.2. Show that $\mathsf{mean}\,(\{x + c\}) = \mathsf{mean}\,(\{x\}) + c$ by substituting into the definition.
2.3. Show that $\sum_{i=1}^{N}(x_i - \mathsf{mean}\,(\{x\})) = 0$ by substituting into the definition.
2.4. Show that $\mathsf{std}\,(x + c) = \mathsf{std}\,(x)$ by substituting into the definition (you'll need to recall the properties of the mean to do this).
2.5. Show that $\mathsf{std}\,(kx) = k\mathsf{std}\,(x)$ by substituting into the definition (you'll need to recall the properties of the mean to do this).
2.6. Show that $\mathsf{median}\,(\{x + c\}) = \mathsf{median}\,(\{x\}) + c$ by substituting into the definition.
2.7. Show that $\mathsf{median}\,(\{kx\}) = k\mathsf{median}\,(\{x\})$ by substituting into the definition.
2.8. Show that $\mathsf{iqr}\{x + c\} = \mathsf{iqr}\{x\}$ by substituting into the definition.
2.9. Show that $\mathsf{iqr}\{kx\} = k\mathsf{iqr}\{x\}$ by substituting into the definition.

PROGRAMMING EXERCISES

2.10. You can find a data set showing the number of barrels of oil produced, per year for the years 1880-1984 at `http://lib.stat.cmu.edu/DASL/Datafiles/Oilproduction.html`. Is a mean a useful summary of this dataset? Why?
2.11. You can find a dataset giving the cost (in 1976 US dollars), number of megawatts, and year of construction of a set of nuclear power plants at `http://lib.stat.cmu.edu/DASL/Datafiles/NuclearPlants.html`.
(a) Are there outliers in this data?
(b) What is the mean cost of a power plant? What is the standard deviation?
(c) What is the mean cost per megawatt? What is the standard deviation?
(d) Plot a histogram of the cost per megawatt. Is it skewed? Why?
2.12. You can find a dataset giving the sodium content and calorie content of three types of hot dog at `http://lib.stat.cmu.edu/DASL/Datafiles/Hotdogs.html`. The types are Beef, Poultry, and Meat (a rather disturbingly vague label). Use class-conditional histograms to compare these three types of hot dog with respect to sodium content and calories.
2.13. You will find a dataset giving (among other things) the number of 3 or more syllable words in advertising copy appearing in magazines at `http://lib.stat.cmu.edu/DASL/Datafiles/magadsdat.html`. The magazines are grouped by the education level of their readers; the groups are 1, 2, and 3 (the variable is called GRP in the data).
(a) Use a boxplot to compare the number of three or more syllable words for the ads in magazines in these three groups. What do you see?

(b) Use a boxplot to compare the number of sentences appearing in the ads in magazines in these three groups. What do you see?

2.14. You can find a dataset giving various properties of public colleges ranked by Kiplingers at `http://www.kiplinger.com/tool/college/T014-S001-kiplinger-s-best-values-in-publi index.php`. To obtain the data, just select the area on your web browser and paste it into an editor window. I had to then do a little fooling around to remove commas in long numbers, get rid of $ signs and the like, but then I had a comma separated list. Similar data for private colleges can be found at `http://www.kiplinger.com/tool/college/T014-S001-kiplinger-s-best-values-in-private-colleges/index.php`. You should look at the R code fragment 3.13 to get started.

(a) If you run that fragment on your files, then look at the figures, they should suggest that there isn't much difference in student debt between top, mid and bottom third public colleges. But there is for private colleges. What happens if you subdivide the top third into smaller pieces? What do you think is going on here?

(b) If you run that fragment on your files, then look at the figures, they should suggest that there isn't much difference in faculty per student between top, mid and bottom third public colleges. But there is for private colleges. What happens if you subdivide the top third into smaller pieces? What do you think is going on here?

(c) Use a set of box plots to come to conclusions about how cost varies with quality for the private colleges.

(d) Use a set of box plots to come to conclusions about how cost varies with quality for the public colleges.

CHAPTER 3

Intermezzo - Programming Tools

This book is not really about how to program. While I'm assuming you know a bit about how to program, quite a lot of what we'll do is use various tricks to avoid programming. We could: use packages; exploit other people's programs; or bolt together a bunch of different programs with scripts, etc. Generally, it's quite useful to know a programming environment that has lots of useful code that other people have written and published that you can use. The environment should have lots of code available to: read datasets in a variety of formats; manipulate them a bit (for example, drop items with missing values); compute useful properties of data; and produce a wide variety of graphs and plots. It should be relatively easy to learn, with lots of tutorials and books available.

You should not feel the need to become an expert in using this programming environment. You just need to be able to cook up strategies to make it do what you want to do. There are two environments that I use regularly, and advocate for use with this book. That said, I'm not going to give you extensive instruction in how to use these environments. Instead, there is some information, some sample code, and a lot of pointers to tutorials, etc.

3.1 MATLAB

Matlab is a programming environment widely used in numerical analysis and computer vision circles, among other. I will use Matlab in examples here, because I'm fairly fluent in Matlab, and give some guidelines. Many universities, including UIUC, have free student licenses for Matlab, and there are numerous reference books and tutorials you can look at for details of syntax, etc. Matlab's focuses on representations of vectors and matrices. The syntax is rather like Fortran, except that there are some special matrix and vector operations. Fluent Matlab programmers replace iterations with these operations to get faster code; some people are very good at this sort of trick.

An alternative widely used in statistical circles is R, which is an open-source language rather like S (a paid license statistics package). I didn't know much R at start of writing, and haven't managed to pick up the kind of fluency required to prepare figures in it, so I won't show R code here. But you should be aware of it as an alternative.

If you want to know what a Matlab command does, you can use `help` or `doc`. For example, you could type `doc sum`. In this chapter, I'll show code I used to make some of the figures and examples in chapter one, to give you some examples for Matlab code.

Listing 3.1 shows how I made the cheese histogram of figure 2.2. Notice I couldn't be bothered to write a reader for this dataset. Instead, I downloaded the text file, then used an editor to cut out the numbers and make them into a constant

Listing 3.1: Matlab code used to make the cheese histogram in figure 2.2

```
cheeses=[12.3, 20.9, 39, 47.9, 5.6, 25.9, ...
    37.3, 21.9, 18.1, 21, 34.9, 57.2, 0.7, ...
    25.9, 54.9, 40.9 , 15.9, 6.4, 18, 38.9, ...
    14, 15.2, 32, 56.7, 16.8, 11.6, 26.5, ...
    0.7 , 13.4 , 5.5];
figure(1);
hist(cheeses, [5, 15, 25, 35, 45, 55])
figure(1);
axis([0, 70, 0, 15])
ylabel('Number of data items');
xlabel('Cheese goodness, in cheese goodness units');
title('Histogram of cheese goodness score for 30 cheeses');
set(gca, 'XTick', [0:7]*10);
set(gca, 'YTick', [0:2:15]);
figure(1);
```

vector. This isn't the best approach for a big file or a general problem, but if you want to take a quick look at something small it might be a sensible thing to do. I then used `figure` to make a window to draw into. The '(1)' argument allows me to keep track of the figure and add things to it; if I hadn't called figure, Matlab would have produced a window, but I'd have to do some work if I wanted to add to it. The command `hist` produces a histogram of a vector (you can look at the `doc` for an explanation of the argument), and then I added labels and a title.

I have listed useful commands for computing summaries below. To give you an example of normalizing data, there is listing 3.2. Notice how I changed the fonts for the axes. I did this to give me a figure I could print, but you may not need to do this.

Matlab will do boxplots, too. In listing 3.3, I show the Matlab code I used to make Figure 2.11.

Boxplots can get pretty elaborate, and putting the right marker in the right place gets interesting. I show the code I used to make Figure 2.13, which required a certain amount of fooling around with strings to get the right marker and put it in the right place.

Listings 3.6 and **??** show how I made the bar chart of genders in Figure 2.1. There are some points to notice here. I cut-and-pasted the data from the web page (`http://lib.stat.cmu.edu/DASL/Datafiles/PopularKids.html`), put the result into a spreadsheet to make a .xls file, saved that, then read it into Matlab using `xlsread`. I did this because it meant I didn't have to write any kind of file reader. You can learn more about `xlsread` by doing `doc xlsread`. It isn't perfect — or, at least, the one on my Mac isn't — because it doesn't like some kinds of format, and it seems not to like very long files. It was enough for this purpose. It produces a cell array (`doc cell`) for the text fields, and an ordinary array for the numeric fields. The cell array is an array whose entries can contain objects like strings, etc. You will find the question of how to get a dataset from the form in which you find it into a form you want in your programming environment to be a persistent nuisance. It's usually approached with quick-and-dirty solutions like this

Listing 3.2: Matlab code used to make the normalized oyster volume histogram in figure 2.7

```
% Change default text fonts for a clean figure.
% You may not really need to do this.
set(0,'DefaultAxesFontName', 'Times New Roman')
set(0,'DefaultAxesFontSize', 28)
set(0,'DefaultTextFontname', 'Times New Roman')
set(0,'DefaultTextFontSize', 28)

cd('~/Current/Courses/Probcourse/SomeData/DataSets/');
% This is where I keep the file
[num2, txt, raw]=xlsread('30oystersdata.xls');
% This data is stored as:
%   index weight volume something somethingelse
wdat=num2(:, 4);
% the volume field turns up here
wm=mean(wdat);
wsd=std(wdat);
figure(1);
hist((wdat-wm)/wsd);
figure(1)
axis([-6 6 0 5])
% sets the axes of the plot
title('Volumes of oysters, standard coordinates');
% Now put this figure in a file
print -depsc2 noysters.eps
```

one.

I have cut from this file the code that I used to get the fonts on the captions right; it takes up space, and doesn't have much to offer. You should have noticed one nice thing. Once I've actually gotten my data into a reasonable form, making the bar chart is easy (`bar` does the trick). It's more trouble to get the title on the figure and the fonts right, etc. than to make the figure. This is true for histograms, as well (listing 2.2). In this case, the data was just a vector, so I cut it out from the web page and stuck it into the code. Look at `doc hist`, and notice you can

Listing 3.3: Matlab code used to make the boxplot in figure 2.11

```
% pizza size data
set(0,'DefaultAxesFontName', 'Times New Roman')
set(0,'DefaultAxesFontSize', 11)
% Change default text fonts.
set(0,'DefaultTextFontname', 'Times New Roman')
set(0,'DefaultTextFontSize', 11)
cd('~/Current/Courses/Probcourse/SomeData/DataSets/');
[num, txt, raw]=xlsread('cleanpizzasize.xls');
ndat=size(num, 1);
figure(1); boxplot(num(:, 5), txt(:, 2)); figure(1)
title('Box plots of pizzas by maker');
print -depsc2 pmakerboxes.eps
```

Listing 3.4: Matlab code used to make the boxplot in figure 2.13

```
% pizza size data
set(0,'DefaultAxesFontName', 'Times New Roman')
set(0,'DefaultAxesFontSize', 11)
% Change default text fonts.
set(0,'DefaultTextFontname', 'Times New Roman')
set(0,'DefaultTextFontSize', 11)
cd('~/Current/Courses/Probcourse/SomeData/DataSets/');
[num, txt, raw]=xlsread('cleanpizzasize.xls');
ndat=size(num, 1);
t2=txt(:, 1);
for i=1:ndat
    foo=txt(i, 2); bar=foo{1}; c1=bar(1);
    % this gets a letter for the maker
    foo=txt(i, 3); bar=foo{1}; c2=bar(1);
    % this gets a letter for the crust
    foo=txt(i, 4); bar=foo{1}; c3=bar(1);c4=bar(end);
    % this gets first, last letter for the topping
    t2(i)={strcat(c1,  c2, c3, c4)};
end
figure(2); boxplot(num(:, 5), t2, 'grouporder', {'DCBs','DCHn',...
    'DCSe', 'DDBs', 'DDHn', 'DDSe', 'DTBs', 'DTHn', 'DTSe', ...
    'EDBs', 'EDHn', 'EDSo', 'EMBs', 'EMHn', 'EMSo', 'ETBs', ...
    'ETHn', 'ETSo'}); figure(2);
title('Box plots of pizzas by maker, type, and topping');
```

change the number of bars and also the centers of the bars if you wish.

Scatter plots are easy in matlab. Listing 4.14 shows the code I used to make the scaled figure in Figure 4.14. In the original data, prices are in shillings and pence. I reduced this to pennies by multiplying the shillings by twelve, and adding the number of pence.

Series are also straightforward to plot. Listing 3.8 shows the code I used to make one of the plots in Figure 4.5. The data has two series — one before an intervention, and one after. I plotted the before in the figure, but the code shows how to plot both. `plot` has a variety of options and tricks; look at `doc plot` for details.

Some useful commands:

- `bar` makes a bar chart.
- `hist` makes a histogram.
- `xlsread` will read many spreadsheet files.
- `mean` computes the mean of a dataset represented as a column vector. If you give it an array, it will compute the mean of each column.
- `std` computes the standard deviation of a dataset represented as a column vector. If you give it an array, it will compute the standard deviation of each column.

Listing 3.5: Matlab code used to read data for the gender bar chart and pie chart in figure 2.1

```
cd('~/Current/courses/ProbCourse/SomeData/MatlabCode');
[num, txt]=xlsread('../Datasets/schooldata.xls');
wv=zeros(size(num, 1), 1);
wv2=zeros(size(num, 1), 1);
% next lines form a really simple parser
for i=1:size(num, 1)
    if strcmp(txt(i, 7), 'Sports')
        wv(i)=1;
    elseif strcmp(txt(i, 7), 'Grades')
        wv(i)=2;
    elseif strcmp(txt(i, 7), 'Popular')
        wv(i)=3;
    end
    if strcmp(txt(i, 1), 'boy')
        wv2(i)=1;
    elseif strcmp(txt(i, 1), 'girl')
        wv2(i)=2;
    end
end
```

- `var` computes the variance of a dataset represented as a column vector. If you give it an array, it will compute the variance of each column.
- `median` computes the median of a dataset represented as a column vector. If you give it an array, it will compute the median of each column.
- `prctile` can compute a set of percentiles for a dataset represented as a column vector. If you give it an array, it will compute those percentiles for each column. Note it takes two argument; you should do `doc prctile`.
- `boxplot` will produce a boxplot; there are many arguments and tricks, so you should do `doc boxplot`.

3.2 R

R is a free programming environment that is very widely used by statisticians and data analysts. R has an extremely rich set of tools for reading, analyzing, and plotting data. I have found R very good for taking quick looks at datasets, even though I'm not a fluent R programmer. My experience has been that I can usually modify some R programs that I already have to get what I want, rather than write a new one (which I find hard, because I keep forgetting details of syntax). I have found that, if there is a statistical algorithm I'd like to try, I can usually fairly easily find at least one implementation in R that I can fiddle with.

History of R: The wikipedia page `http://en.wikipedia.org/wiki/R_(programming_language)` has some information on the history and implementation of R.

Obtaining and installing R: The main page for R is at `http://www.r-project.org`, where (unless you have a quite unusual computer system) you can find a set of binaries for your system. If you have the urge to compile it

Listing 3.6: Matlab code used to produce plots for the gender bar chart and pie chart in figure 2.1

```
cvec=zeros(6, 1);
cvec(3)=sum((wv==1).*(wv2==1));
cvec(2)=sum((wv==2).*(wv2==1));
cvec(1)=sum((wv==3).*(wv2==1));
cvec(4)=sum((wv==1).*(wv2==2));
cvec(5)=sum((wv==2).*(wv2==2));
cvec(6)=sum((wv==3).*(wv2==2));
% cvec now contains counts of each case
figure(6)
pie(cvec, {'boy-Popular', 'boy-Grades', 'boy-Sports', ...
    'girl-Sports', 'girl-Grades', 'girl-Popular'});
title('Number of each gender choosing each goal');
print -depsc2 ../Figures/childpiegendergoal.eps
%%
figure(7)
bar(cvec);
set(gca, 'XTick', [1, 2, 3, 4, 5, 6]);
set(gca, 'XTickLabel',{'b-P', 'b-G', 'b-S', 'g-S', 'g-G', 'g-P'});
figure(7);
print -depsc2 ../Figures/childbargendergoal.eps
```

yourself, you can also find source and some information about how to make R on some systems. If you're having trouble installing R, you should look at the FAQ at `http://www.r-project.org`, which has entries like "How can R be obtained?" and "How can R be installed?".

Learning to use R: There are a variety of books and web pages on how to use R. A good introduction is at `http://cran.r-project.org/doc/manuals/r-release/R-intro.html`. There is a list of learning resources at `http://www.ats.ucla.edu/stat/r/`. If you look at `http://www.ats.ucla.edu/stat/r/seminars/intro.htm`, you will find a set of slides that give an introduction to R; at `\http://www.ats.ucla.edu/stat/r/seminars/intro.R`, you will find a listing of the code used in the slides. If you'd like a book, then you could search Amazon with the keyword R, then read the reviews. I have a hard time recommending any one book. I have bought several, at various levels, and have found each a bit helpful for some things, but there is no book I always reach for first. I have found it sufficient to have quite a shallow understanding of R, and I can usually get things done by copying and then modifying existing code.

3.2.1 R examples

If you don't know what an R function does, do `?hist` (to get information on `hist`). This should throw up a help window. If it doesn't, `??hist` is sometimes helpful. If I can't understand the help window, which happens more often than I care to admit, I search the web.

Listing 3.9 shows how I could have made the cheese histogram of figure 2.2 using R. Notice I couldn't be bothered to write a reader for this dataset. Instead, I downloaded the text file, then used an editor to cut out the numbers and make

Listing 3.7: Matlab code used to make the scatter plot in figure 4.14

```
set(0,'DefaultAxesFontName', 'Times New Roman')
set(0,'DefaultAxesFontSize', 28)
% Change default text fonts.
set(0,'DefaultTextFontname', 'Times New Roman')
set(0,'DefaultTextFontSize', 28)
cd('~/Current/Courses/Probcourse/SomeData/DataSets/');
% this is where I keep the file, but you may have it somewhere else
[num, txt, raw]=xlsread('lynxdata.xls');
pennyprices=12*num(:, 3)+num(:, 4);
mnp=mean(num(1:44, 2));
snp=std(num(1:44, 2));
mpp=mean(pennyprices(1:44));
spp=std(pennyprices(1:44));
figure(3);
plot((num(1:44, 2)-mnp)/snp, (pennyprices(1:44)-mpp)/spp, 'r*');
figure(3)
xlabel('normalized number of pelts');
ylabel('normalized price')
%title('Normalized scatter plot of lynx pelts against price');
print -depsc2 lynxscaledscatter.eps
```

Listing 3.8: Matlab code used to make the Hyde Park series plot in figure 4.5

```
cd('~/Current/Courses/Probcourse/SomeData/DataSets/');
set(0,'DefaultAxesFontName', 'Times New Roman')
set(0,'DefaultAxesFontSize', 28)
% Change default text fonts.
set(0,'DefaultTextFontname', 'Times New Roman')
set(0,'DefaultTextFontSize', 28)
% this is where I keep the file, but you may have it somewhere else
[num, txt, raw]=xlsread('hydeparkburglaries.xls');
for i=1:size(num, 1)-1
    if (num(i, 2)==0)&&(num(i+1, 2)==1)
        brk=i;
    end
end
prev=num(1:brk, 1);
post=num(brk+1:end, 1);
figure(1); plot(prev, '-o'); figure(1)
xlabel('month');
ylabel('number of burglaries');
title('Burglaries each month in Hyde Park');
axis([0 45 0 110])
print -depsc2 hydeparkburglaries.eps
%%
figure(2); clf; hold off
plot([1:brk], prev, '-o', [brk+1:size(num, 1)], post, '-rs'); figure(2)
legend('Before intervention', 'After intervention');
xlabel('month');
ylabel('number of burglaries');
title('Number of burglaries each month in Hyde Park, a suburb of Chicago');
%%
axis([0 60 0 110])
print -depsc2 inthydeparkburglaries.eps
```

Listing 3.9: R code for a cheese histogram like in figure 2.2

```
cheeses<- c(12.3, 20.9, 39, 47.9, 5.6, 25.9,
    37.3, 21.9, 18.1, 21, 34.9, 57.2, 0.7,
    25.9, 54.9, 40.9 , 15.9, 6.4, 18, 38.9,
    14, 15.2, 32, 56.7, 16.8, 11.6, 26.5,
    0.7 , 13.4 , 5.5)
cbr<-seq(0, 60, by=10)
chh<-hist(cheeses , breaks=cbr)
```

Listing 3.10: R code used to make a figure like the normalized oyster volume histogram in figure 2.7

```
setwd('~/Current/Courses/Probcourse/SomeData/DataSets/')
oysters<-read.csv('30oystersdata.csv', header=FALSE);
summary(oysters) # not strictly necessary
oystervolumes<-oysters$V3
omean<-mean(oystervolumes)
zmo<-oystervolumes-omean
osd<-sqrt(sum(zmo^2)/length(zmo))
normoysters<-zmo/osd
hist(normoysters, breaks=seq(-6, 6, by=0.5));
```

them into a constant vector. This isn't the best approach for a big file or a general problem, but if you want to take a quick look at something small it might be a sensible thing to do. Notice how to make a constant vector in R. I've told R what bins to use (the line using `seq` produces a sequence of values, which gets passed in using the `breaks` argument). `hist` is (like most R commands) rather rich in options; I found `http://www.r-bloggers.com/basics-of-histograms/` helpful when I wanted to understand some of them.

R usually just then throws a window on your screen with the figure in it. If you want a file you can incorporate into a document or print, you have to do more. This is illustrated in the next example. I didn't fiddle with the labels, etc. because I wanted a simple example.

I show the code for producing a figure rather like the normalized oyster histogram of figure 2.7 in listing 3.10. Because I don't have a good reader for Xcel files in R, I export the files to a comma separated value format, then use the R function `read.csv`. In this case, there was no header in the data, and I told R that. R produces a fairly rich data structure called a data table, but I only really cared about the volume column, so I extract that to produce `oystervolumes`. R has a function to compute the standard deviation, but irritatingly it isn't the standard deviation we want (see the point of confusion on 22). The code shows how I computed the value I need, then set up a histogram with a reasonable set of breaks.

One useful point: once you've read a dataset into R, it's usually a good idea to use `summary`. This gives you some information about the data set. It isn't necessary for producing a histogram, but it's informative, which is why I put it into the listing.

R will do boxplots, too. In listing 3.11, I show R code to make a figure

Listing 3.11: R code used to make a boxplot like that in figure 2.11

```
setwd('~/Current/Courses/Probcourse/SomeData/DataSets/')
pizza<-read.csv('cleanpizzasize.csv', header=FALSE)
boxplot(V5~V2, data=pizza)
```

Listing 3.12: R code for more boxplots, illustrating a neat trick with data tables.

```
setwd('~/Current/Courses/Probcourse/SomeData/DataSets/')
pizza<-read.csv('cleanpizzasize.csv', header=FALSE)
boxplot(V5~V2+V3, data=pizza)
```

like Figure 2.11. There is one bit of R deviousness here, which I've expanded in following listings. A data table is an object with quite a lot of structure. The version of `boxplot` I've used here says, in effect: "prepare a boxplot of the values of V5 for each of the cases in V2 taken from the data table pizza". V2 takes the values "Dominos" and "EagleBoys", so I get one boxplot for each manufacturer.

This trick can be extended. Listing 3.12 shows R code that does one box per manufacturer and crust type. The version of `boxplot` I've used here says, in effect: "prepare a boxplot of the values of V5 for each of the cases in V2 and V3 taken from the data table pizza". V2 takes the values "Dominos" and "EagleBoys", and V3 has each crust type.

If you fire up R and produce the figure, you should be a bit unhappy with the labelling of the boxes. You'll need to do some work to fix this.

Listing 3.13 may help you get started on one of the exercises.

Listing 3.14 illustrates some other tricks with R. In particular, getting counts of a categorical variable is easy with `table`. I normalized the tables by iterating with `for`, and there is some code that shows how to print a postscript figure to a file.

Scatter plots are easy in R. Listing 4.14 shows the code I used to make the

Listing 3.13: R code for college examples.

```
setwd('~/Current/Courses/Probcourse/SomeData/DataSets/')
publics<-read.csv("publics.csv",header=FALSE)
publics$V12<-cut(publics$V1, breaks=c(0, 33, 66, 100))
# this creates three equivalence classes:
# top third, mid third and bottom third
# then sticks them back into the data frame
boxplot(V11~V12, data=publics)
# now we have a box plot of debt for each third
boxplot(V5~V12, data=publics)
# now we have a box plot of number of faculty for each third
privates<-read.csv("privates.csv",header=FALSE)
privates$V12<-cut(privates$V1, breaks=c(0, 33, 66, 100))
boxplot(V11~V12, data=privates)
boxplot(V5~V12, data=privates)
```

Listing 3.14: R code for bar charts of figure 3.14.

```
setwd('~/Current/courses/ProbCourse/SomeData/Datasets');
schooldata<-read.csv('schooldata.csv', header=FALSE);
counts<-table(schooldata$V1, schooldata$V7)
setEPS()
postscript("goalsbygender.eps")
barplot(counts, main="Goals by gender",
  xlab="Goals", col=c("darkblue","red"),
          legend = rownames(counts),xpd=FALSE, xlim=c(0, 100), width=20)
dev.off()
counts2<-table(schooldata$V7, schooldata$V1)
setEPS()
postscript("genderbygoals.eps")
barplot(counts2, main="Gender by goals",
  xlab="Gender", col=c("darkblue","red", "green"),
          legend = colnames(counts), xpd=FALSE, xlim=c(0, 100), width=20)
dev.off()
counts3<-counts2
for (i in 1:2) {
  counts3[,i]<-counts3[,i]/sum(counts3[,i])
}
setEPS()
postscript("genderbygoalsrelfreq.eps")
barplot(counts3, main="Gender by goals, relative frequencies",
  xlab="Gender", col=c("darkblue","red", "green"),
          legend = colnames(counts), xpd=FALSE, xlim=c(0, 100), width=20)
dev.off()
counts4<-counts
for (i in 1:3) {
  counts4[,i]<-counts4[,i]/sum(counts4[,i])
}
setEPS()
postscript("goalsbygenderrelfreq.eps")
barplot(counts4, main="Goals by gender, relative frequencies",
  xlab="Goals", col=c("darkblue","red"),
          legend = rownames(counts),xpd=FALSE, xlim=c(0, 100), width=20)
dev.off()
```

scaled figure in Figure 4.14. In the original data, prices are in shillings and pence. I reduced this to pennies by multiplying the shillings by twelve, and adding the number of pence. Notice I haven't used R's `sd` function, because it isn't the standard deviation we want (see the point of confusion on 22).

Series are also straightforward to plot. Listing 3.16 shows the code I used to make a plot like one of the plots in Figure 4.5. The data has two series — one before an intervention, and one after. Instead of walking along the data in my code (like I did in the Matlab code for this example), I cut out the first series by hand and pasted it into a csv file. This made it easy to plot the before series in the figure. You should be constantly looking out for tricks to make data handling easier; we're typically not looking for a clean programming solution, just a way to make some problems go away.

Listing 3.15: R code used to make a scatter plot like that of figure 4.14

```
setwd('~/Current/Courses/Probcourse/SomeData/DataSets/');
lynxdata<-read.csv('lynxdata.csv', header=FALSE);
lynxdata$V5=12*lynxdata$V3+lynxdata$V4
# this gives prices in pennies
# interested in years 1-44
lpen<-lynxdata$V5[1:44]
lpm<-mean(lpen)
lps<-sqrt((sum((lpen-lpm)^2))/length(lpen))
lpn<-(lpen-lpm)/lps
lnums<-lynxdata$V1[1:44]
lnm<-mean(lnums)
lns<-sqrt((sum((lnums-lnm)^2))/length(lnums))
lnn<-(lnums-lnm)/lns
plot(lnn, lpn, xlab="Number of pelts", ylab="Price in pennies", asp=1)
```

Listing 3.16: R code used to make a series like the Hyde Park series plot in figure 4.5

```
setwd('~/Current/Courses/Probcourse/SomeData/DataSets/')
hydepark<-read.csv('hydeparksmall.csv', header=FALSE)
plot(hydepark$V1, type='b', ylab="Number of burglaries", xlab="Days")
```

CHAPTER 4

Looking at Relationships

We think of a dataset as a collection of d-tuples (a d-tuple is an ordered list of d elements). For example, the Chase and Dunner dataset had entries for Gender; Grade; Age; Race; Urban/Rural; School; Goals; Grades; Sports; Looks; and Money (so it consisted of 11-tuples). The previous chapter explored methods to visualize and summarize a set of values obtained by extracting a single element from each tuple. For example, I could visualize the heights or the weights of a population (as in Figure 2.7). But I could say nothing about the relationship between the height and weight. In this chapter, we will look at methods to visualize and summarize the relationships between pairs of elements of a dataset.

4.1 PLOTTING 2D DATA

We take a dataset, choose two different entries, and extract the corresponding elements from each tuple. The result is a dataset consisting of 2-tuples, and we think of this as a two dimensional dataset. The first step is to plot this dataset in a way that reveals relationships. The topic of how best to plot data fills many books, and we can only scratch the surface here. Categorical data can be particularly tricky, because there are a variety of choices we can make, and the usefulness of each tends to depend on the dataset and to some extent on one's cleverness in graphic design (section 4.1.1).

For some continuous data, we can plot the one entry as a function of the other (so, for example, our tuples might consist of the date and the number of robberies; or the year and the price of lynx pelts; and so on, section 4.1.2).

Mostly, we use a simple device, called a scatter plot. Using and thinking about scatter plots will reveal a great deal about the relationships between our data items (section 4.1.3).

4.1.1 Categorical Data, Counts, and Charts

Categorical data is a bit special. Assume we have a dataset with several categorical descriptions of each data item. One way to plot this data is to think of it as belonging to a richer set of categories. Assume the dataset has categorical descriptions, which are not ordinal. Then we can construct a new set of categories by looking at each of the cases for each of the descriptions. For example, in the Chase and Dunner data of table 2.2, our new categories would be: "boy-sports"; "girl-sports"; "boy-popular"; "girl-popular"; "boy-grades"; and "girl-grades". A large set of categories like this can result in a poor bar chart, though, because there may be too many bars to group the bars successfully. Figure 4.1 shows such a bar chart. Notice that it is hard to group categories by eye to compare; for example, you can see that slightly more girls think grades are important than boys do, but to do so you need to compare two bars that are separated by two other bars. An alternative is a **pie**

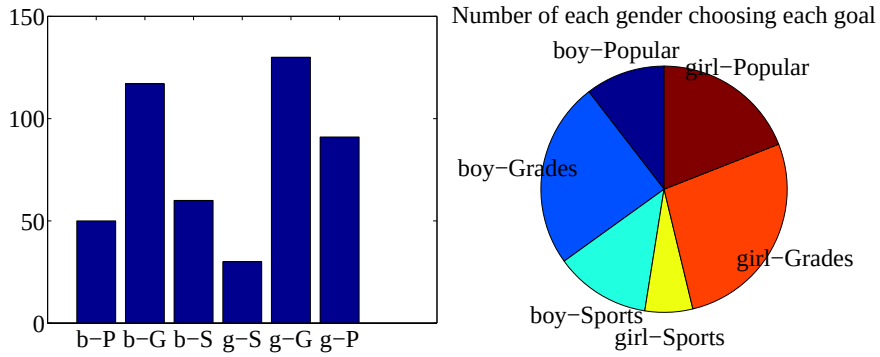

FIGURE 4.1: *I sorted the children in the Chase and Dunner study into six categories (two genders by three goals), and counted the number of children that fell into each cell. I then produced the bar chart on the* **left**, *which shows the number of children of each gender, selecting each goal. On the* **right**, *a pie chart of this information. I have organized the pie chart so it is easy to compare boys and girls by eye — start at the top; going down on the left side are boy goals, and on the right side are girl goals. Comparing the size of the corresponding wedges allows you to tell what goals boys (resp. girls) identify with more or less often.*

chart, where a circle is divided into sections whose angle is proportional to the size of the data item. You can think of the circle as a pie, and each section as a slice of pie. Figure 4.1 shows a pie chart, where each section is proportional to the number of students in its category. In this case, I've used my judgement to lay the categories out in a way that makes comparisons easy. I'm not aware of any tight algorithm for doing this, though.

Pie charts have problems, because it is hard to judge small differences in area accurately by eye. For example, from the pie chart in figure 4.1, it's hard to tell that the "boy-sports" category is slightly bigger than the "boy-popular" category (try it; check using the bar chart). For either kind of chart, it is quite important to think about *what* you plot. For example, the plot of figure 4.1 shows the total number of respondents, and if you refer to figure 2.1, you will notice that there are slightly more girls in the study. Is the *percentage* of boys who think grades are important smaller (or larger) than the *percentage* of girls who think so? you can't tell from these plots, and you'd have to plot the percentages instead.

An alternative is to use a **stacked bar chart**. You can (say) regard the data as of two types, "Boys" and "Girls". Within those types, there are subtypes ("Popularity", "Grades" and "Sport"). The height of the bar is given by the number of elements in the type, and the bar is divided into sections corresponding to the number of elements of that subtype. Alternatively, if you want the plot to show relative frequencies, the bars could all be the same height, but the shading corresponds to the fraction of elements of that subtype. This is all much harder to say than to see or to do (Figure 4.2).

An alternative to a pie chart that is very useful for two dimensional data is a **heat map**. This is a method of displaying a matrix as an image. Each entry of

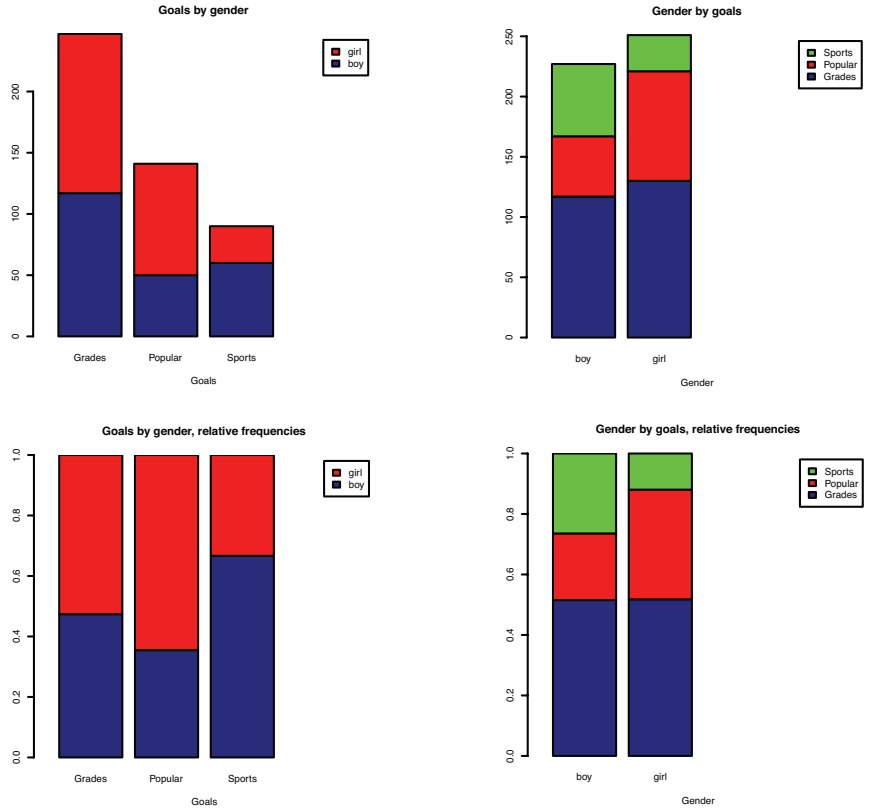

FIGURE 4.2: *These bar charts use stacked bars. In the* **top row***, the overall height of the bar is given by the number of elements of that type but each different subtype is identified by shading, so you can tell by eye, for example, how many of the "Grades" in the study were "Boys". This layout makes it hard to tell what* fraction *of, say, "Boys" aspire to "Popularity". In the* **bottom row***, all bars have the same height, but the shading of the bar identifies the fraction of that type that has a corresponding subtype. This means you can tell by eye what fraction of "Girls" aspire to "Sports".*

the matrix is mapped to a color, and the matrix is represented as an image. For the Chase and Dunner study, I constructed a matrix where each row corresponds to a choice of "sports", "grades", or "popular", and each column corresponds to a choice of "boy" or "girl". Each entry contains the count of data items of that type. Zero values are represented as white; the largest values as red; and as the value increases, we use an increasingly saturated pink. This plot is shown in figure 4.3

If the categorical data is ordinal, the ordering offers some hints for making a good plot. For example, imagine we are building a user interface. We build an initial version, and collect some users, asking each to rate the interface on scales for "ease of use" (-2, -1, 0, 1, 2, running from bad to good) and "enjoyability" (again, -2, -1, 0, 1, 2, running from bad to good). It is natural to build a 5x5 table, where

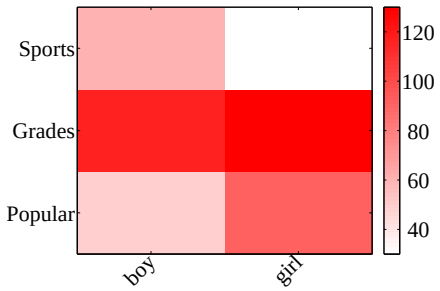

FIGURE 4.3: *A heat map of the Chase and Dunner data. The color of each cell corresponds to the count of the number of elements of that type. The colorbar at the side gives the correspondence between color and count. You can see at a glance that the number of boys and girls who prefer grades is about the same; that about the same number of boys prefer sports and popularity, with sports showing a mild lead; and that more girls prefer popularity to sports.*

	-2	-1	0	1	2
-2	24	5	0	0	1
-1	6	12	3	0	0
0	2	4	13	6	0
1	0	0	3	13	2
2	0	0	0	1	5

TABLE 4.1: *I simulated data representing user evaluations of a user interface. Each cell in the table on the* **left** *contains the count of users rating "ease of use" (horizontal, on a scale of -2 -very bad- to 2 -very good) vs. "enjoyability" (vertical, same scale). Users who found the interface hard to use did not like using it either. While this data is categorical, it's also ordinal, so that the order of the cells is determined. It wouldn't make sense, for example, to reorder the columns of the table or the rows of the table.*

each cell represents a pair of "ease of use" and "enjoyability" values. We then count the number of users in each cell, and build graphical representations of this table. One natural representation is a **3D bar chart**, where each bar sits on its cell in the 2D table, and the height of the bars is given by the number of elements in the cell. Table 4.1 shows a table and figure 4.4 shows a 3D bar chart for some simulated data. The main difficulty with a 3D bar chart is that some bars are hidden behind others. This is a regular nuisance. You can improve things by using an interactive tool to rotate the chart to get a nice view, but this doesn't always work. Heatmaps don't suffer from this problem (Figure 4.4), another reason they are a good choice.

4.1.2 Series

Sometimes one component of a dataset gives a natural ordering to the data. For example, we might have a dataset giving the maximum rainfall for each day of the

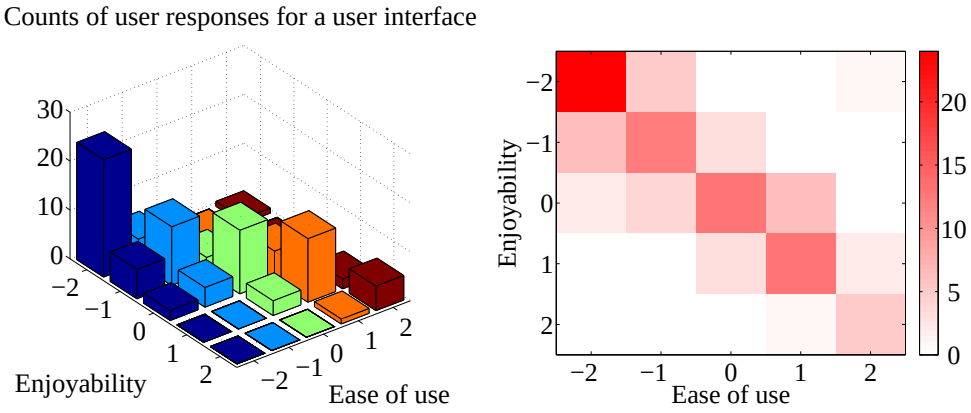

FIGURE 4.4: *On the* **left**, *a 3D bar chart of the data. The height of each bar is given by the number of users in each cell. This figure immediately reveals that users who found the interface hard to use did not like using it either. However, some of the bars at the back are hidden, so some structure might be hard to infer. On the* **right**, *a heat map of this data. Again, this figure immediately reveals that users who found the interface hard to use did not like using it either. It's more apparent that everyone disliked the interface, though, and it's clear that there is no important hidden structure.*

year. We could record this either by using a two-dimensional representation, where one dimension is the number of the day and the other is the temperature, or with a convention where the i'th data item is the rainfall on the i'th day. For example, at `http://lib.stat.cmu.edu/DASL/Datafiles/timeseriesdat.html`, you can find four datasets indexed in this way. It is natural to plot data like this as a function of time. From this dataset, I extracted data giving the number of burglaries each month in a Chicago suburb, Hyde Park. I have plotted part this data in Figure 4.5 (I left out the data to do with treatment effects). It is natural to plot a graph of the burglaries as a function of time (in this case, the number of the month). The plot shows each data point explicitly. I also told the plotting software to draw lines joining data points, because burglaries do not all happen on a specific day. The lines suggest, reasonably enough, the rate at which burglaries are happening between data points.

As another example, at `http://lib.stat.cmu.edu/datasets/Andrews/` you can find a dataset that records the number of lynx pelts traded to the Hudson's Bay company and the price paid for each pelt. This version of the dataset appeared first in table 3.2 of *Data: a Collection of Problems from many Fields for the Student and Research Worker* by D.F. Andrews and A.M. Herzberg, published by Springer in 1985. I have plotted it in figure 4.5. The dataset is famous, because it shows a periodic behavior in the number of pelts (which is a good proxy for the number of lynx), which is interpreted as a result of predator-prey interactions. Lynx eat rabbits. When there are many rabbits, lynx kittens thrive, and soon there will be many lynx; but then they eat most of the rabbits, and starve, at which point

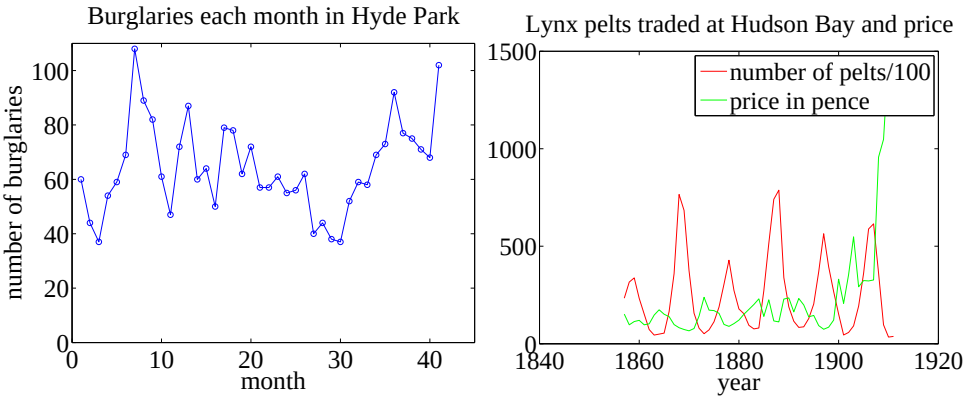

FIGURE 4.5: **Left**, *the number of burglaries in Hyde Park, by month.* **Right**, *a plot of the number of lynx pelts traded at Hudson Bay and of the price paid per pelt, as a function of the year. Notice the scale, and the legend box (the number of pelts is scaled by 100).*

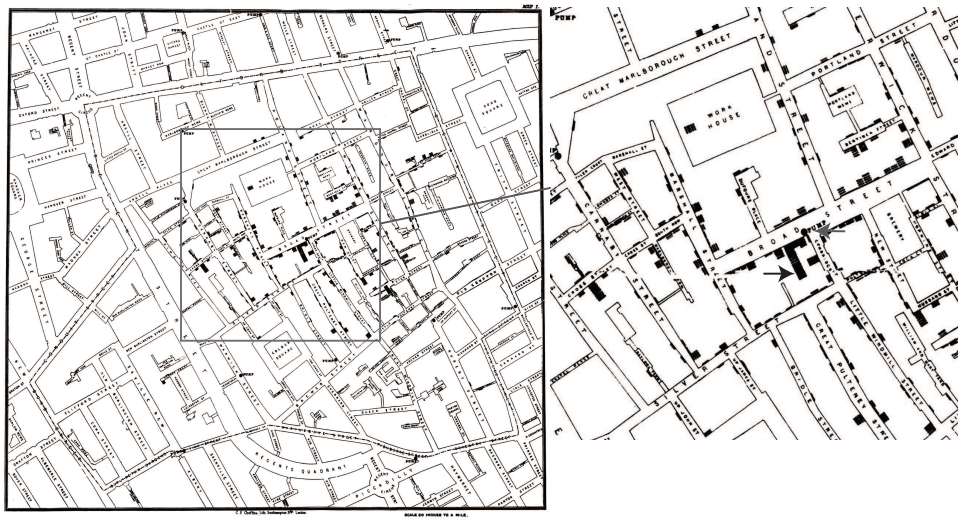

FIGURE 4.6: *Snow's scatter plot of cholera deaths on the* **left**. *Each cholera death is plotted as a small bar on the house in which the bar occurred (for example, the black arrow points to one stack of these bars, indicating many deaths, in the detail on the* **right**). *Notice the fairly clear pattern of many deaths close to the Broad street pump (grey arrow in the detail), and fewer deaths further away (where it was harder to get water from the pump).*

the rabbit population rockets. You should also notice that after about 1900, prices seem to have gone up rather quickly. I don't know why this is. There is also some suggestion, as there should be, that prices are low when there are many pelts, and

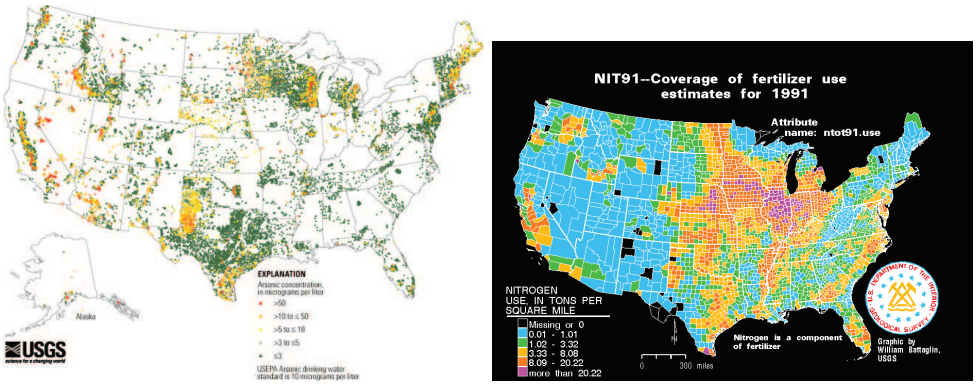

FIGURE 4.7: **Left**, *a scatter plot of arsenic levels in US groundwater, prepared by the US Geological Survey (you can find the data at* `http://water.usgs.gov/GIS/metadata/usgswrd/XML/arsenic_map.xml`*. Here the shape and color of each marker shows the amount of arsenic, and the spatial distribution of the markers shows where the wells were sampled.* **Right**, *the usage of Nitrogen (a component of fertilizer) by US county in 1991, prepared by the US Geological Survey (you can find the data at* `http://water.usgs.gov/GIS/metadata/usgswrd/XML/nit91.xml`*). In this variant of a scatter plot (which usually takes specialized software to prepare) one fills each region with a color indicating the data in that region.*

high when there are few.

4.1.3 Scatter Plots for Spatial Data

It isn't always natural to plot data as a function. For example, in a dataset containing the temperature and blood pressure of a set of patients, there is no reason to believe that temperature is a function of blood pressure, or the other way round. Two people could have the same temperature, and different blood pressures, or vice-versa. As another example, we could be interested in what causes people to die of cholera. We have data indicating *where* each person died in a particular outbreak. It isn't helpful to try and plot such data as a function.

The **scatter plot** is a powerful way to deal with this situation. In the first instance, assume that our data points actually describe points on the a real map. Then, to make a scatter plot, we make a mark on the map at a place indicated by each data point. What the mark looks like, and how we place it, depends on the particular dataset, what we are looking for, how much we are willing to work with complex tools, and our sense of graphic design.

Figure 4.6 is an extremely famous scatter plot, due to John Snow. Snow — one of the founders of epidemiology — used a scatter plot to reason about a cholera outbreak centered on the Broad Street pump in London in 1854. At that time, the mechanism that causes cholera was not known. Snow plotted cholera deaths as little bars (more bars, more deaths) on the location of the house where the death occurred. More bars means more deaths, fewer bars means fewer deaths. There

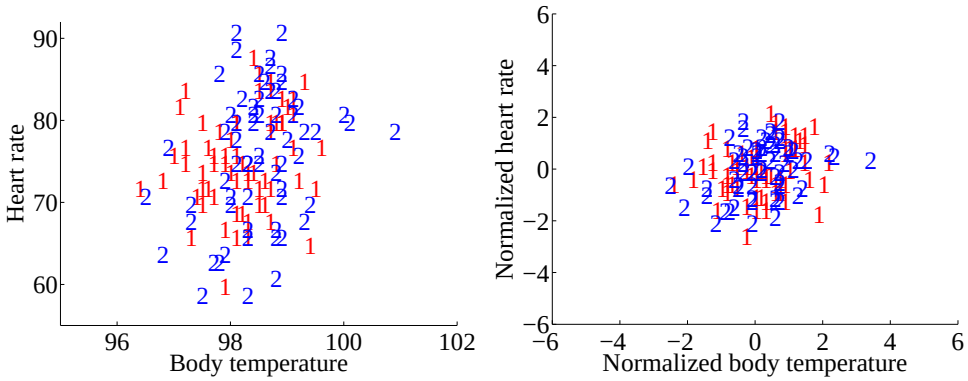

FIGURE 4.8: *A scatter plot of body temperature against heart rate, from the dataset at* `http://www2.stetson.edu/~jrasp/data.htm`*; normtemp.xls. I have separated the two genders by plotting a different symbol for each (though I don't know which gender is indicated by which letter); if you view this in color, the differences in color makes for a greater separation of the scatter. This picture suggests, but doesn't conclusively establish, that there isn't much dependence between temperature and heart rate, and any dependence between temperature and heart rate isn't affected by gender.*

are more bars per block close to the pump, and few far away. This plot offers quite strong evidence of an association between the pump and death from cholera. Snow used this scatter plot as evidence that cholera was associated with water, and that the Broad Street pump was the source of the tainted water.

Figure 4.7 shows a scatter plot of arsenic levels in groundwater for the United States, prepared by the US Geological Survey. The data set was collected by Focazio and others in 2000; by Welch and others in 2000; and then updated by Ryker 2001. It can be found at `http://water.usgs.gov/GIS/metadata/usgswrd/XML/arsenic_map.xml`. One variant of a scatter plot that is particularly useful for geographic data occurs when one fills regions on a map with different colors, following the data in that region. Figure 4.7 shows the nitrogen usage by US county in 1991; again, this figure was prepared by the US Geological Survey.

4.1.4 Exposing Relationships with Scatter Plots

Scatter plots are natural for geographic data, but a scatter plot is a useful, simple tool for ferreting out associations in other kinds of data as well. Now we need some notation. Assume we have a dataset $\{x\}$ of N data items, $x_1, \ldots, x_N$. Each data item is a d dimensional vector (so its components are numbers). We wish to investigate the relationship between two components of the dataset. For example, we might be interested in the 7'th and the 13'th component of the dataset. We will produce a two-dimensional plot, one dimension for each component. It does not really matter which component is plotted on the x-coordinate and which on the y-coordinate (though it will be some pages before this is clear). But it is very difficult to write sensibly without talking about the x and y coordinates.

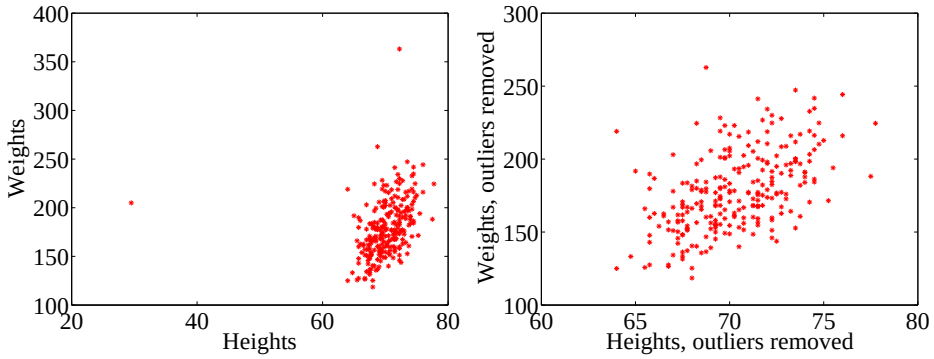

FIGURE 4.9: *A scatter plots of weight against height, from the dataset at* `http://www2.stetson.edu/~jrasp/data.htm`. **Left:** *Notice how two outliers dominate the picture, and to show the outliers, the rest of the data has had to be bunched up.* **Right** *shows the data with the outliers removed. The structure is now somewhat clearer.*

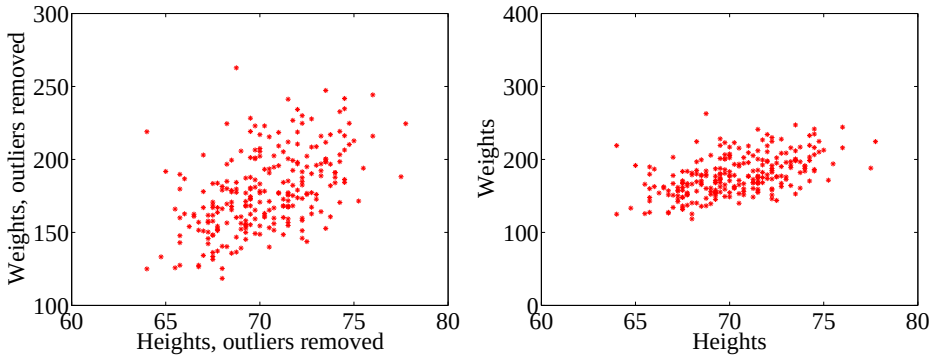

FIGURE 4.10: *Scatter plots of weight against height, from the dataset at* `http://www2.stetson.edu/~jrasp/data.htm`. **Left:** *data with two outliers removed, as in figure 4.9.* **Right:** *this data, rescaled slightly. Notice how the data looks less spread out. But there is no difference between the datasets. Instead, your eye is easily confused by a change of scale.*

We will make a two-dimensional dataset out of the components that interest us. We must choose which component goes first in the resulting 2-vector. We will plot this component on the x-coordinate (and we refer to it as the x-coordinate), and to the other component as the y-coordinate. This is just to make it easier to describe what is going on; there's no important idea here. It really will not matter which is x and which is y. The two components make a dataset $\{\mathbf{x}_i\} = \{(x_i, y_i)\}$. To produce a scatter plot of this data, we plot a small shape at the location of each data item.

Such scatter plots are very revealing. For example, figure 4.8 shows a scatter plot of body temperature against heart rate for humans. In this dataset, the gender

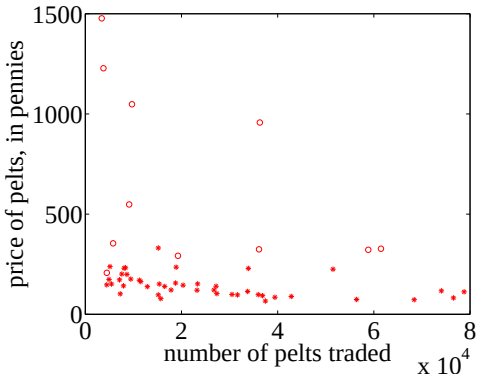

FIGURE 4.11: *A scatter plot of the price of lynx pelts against the number of pelts. I have plotted data for 1901 to the end of the series as circles, and the rest of the data as *'s. It is quite hard to draw any conclusion from this data, because the scale is confusing. Furthermore, the data from 1900 on behaves quite differently from the other data.*

of the subject was recorded (as "1" or "2" — I don't know which is which), and so I have plotted a "1" at each data point with gender "1", and so on. Looking at the data suggests there isn't much difference between the blob of "1" labels and the blob of "2" labels, which suggests that females and males are about the same in this respect.

The scale used for a scatter plot matters. For example, plotting lengths in meters gives a very different scatter from plotting lengths in millimeters. Figure 4.9 shows two scatter plots of weight against height. Each plot is from the same dataset, but one is scaled so as to show two outliers. Keeping these outliers means that the rest of the data looks quite concentrated, just because the axes are in large units. In the other plot, the axis scale has changed (so you can't see the outliers), but the data looks more scattered. This may or may not be a misrepresentation. Figure 4.10 compares the data with outliers removed, with the same plot on a somewhat different set of axes. One plot looks as though increasing height corresponds to increasing weight; the other looks as though it doesn't. This is purely due to deceptive scaling — each plot shows the same dataset.

Dubious data can also contribute to scaling problems. Recall that, in figure 4.5, price data before and after 1900 appeared to behave differently. Figure 4.11 shows a scatter plot of the lynx data, where I have plotted number of pelts against price. I plotted the post-1900 data as circles, and the rest as asterisks. Notice how the circles seem to form a quite different figure, which supports the suggestion that something interesting happened around 1900. The scatter plot does not seem to support the idea that prices go up when supply goes down, which is puzzling, because this is a pretty reliable idea. This turns out to be a scale effect. Scale is an important nuisance, and it's easy to get misled by scale effects. The way to avoid the problem is to plot in standard coordinates.

A natural solution to problems with scale is to normalize the x and y coor-

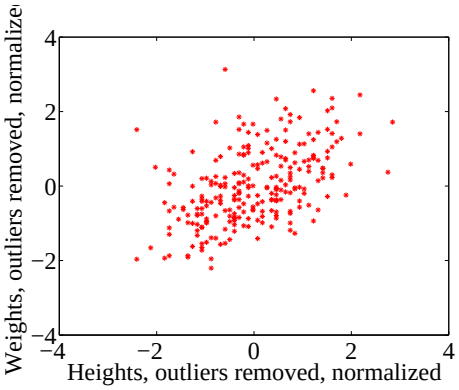

FIGURE 4.12: *A normalized scatter plot of weight against height, from the dataset at* `http://www2.stetson.edu/~jrasp/data.htm`. *Now you can see that someone who is a standard deviation taller than the mean will tend to be somewhat heavier than the mean too.*

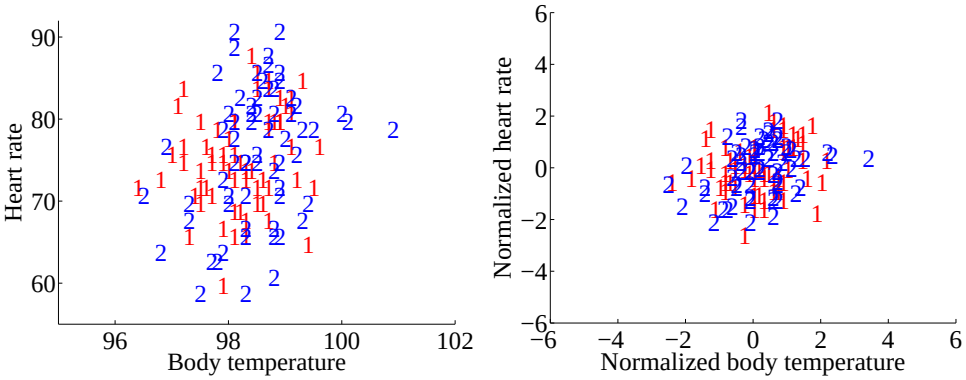

FIGURE 4.13: **Left:** *A scatter plot of body temperature against heart rate, from the dataset at* `http://www2.stetson.edu/~jrasp/data.htm`; *normtemp.xls. I have separated the two genders by plotting a different symbol for each (though I don't know which gender is indicated by which letter); if you view this in color, the differences in color makes for a greater separation of the scatter. This picture suggests, but doesn't conclusively establish, that there isn't much dependence between temperature and heart rate, and any dependence between temperature and heart rate isn't affected by gender. The scatter plot of the normalized data, in standard coordinates, on the* **right** *supports this view.*

dinates of the two-dimensional data to standard coordinates. We can normalize without worrying about the dimension of the data — we normalize each dimension independently by subtracting the mean of that dimension and dividing by the standard deviation of that dimension. We continue to use the convention of writing the normalized x coordinate as $\hat{x}$ and the normalized y coordinate as $\hat{y}$. So, for

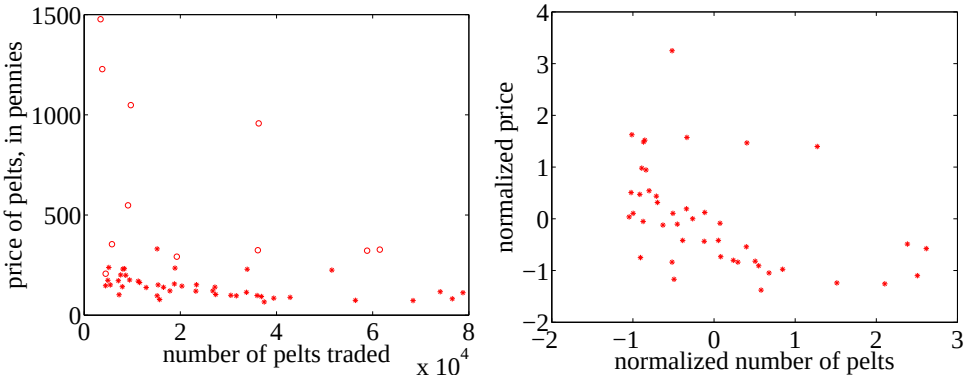

FIGURE 4.14: **Left:** *A scatter plot of the price of lynx pelts against the number of pelts (this is a repeat of figure 4.11 for reference). I have plotted data for 1901 to the end of the series as circles, and the rest of the data as *'s. It is quite hard to draw any conclusion from this data, because the scale is confusing.* **Right:** *A scatter plot of the price of pelts against the number of pelts for lynx pelts. I excluded data for 1901 to the end of the series, and then normalized both price and number of pelts. Notice that there is now a distinct trend; when there are fewer pelts, they are more expensive, and when there are more, they are cheaper.*

example, we can write $\hat{x}_j = (x_j - \mathsf{mean}\,(\{x\}))/\mathsf{std}\,(x))$ for the $\hat{x}$ value of the j'th data item in normalized coordinates. Normalizing shows us the dataset on a standard scale. Once we have done this, it is quite straightforward to read off simple relationships between variables from a scatter plot.

4.2 CORRELATION

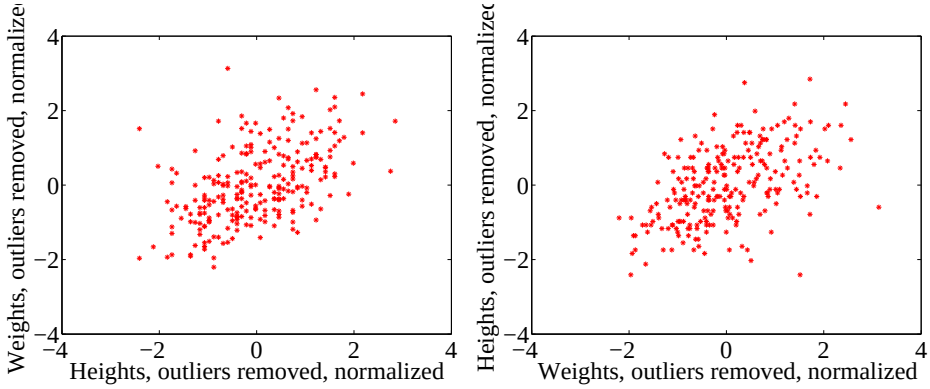

FIGURE 4.15: *On the* **left**, *a normalized scatter plot of weight (y-coordinate) against height (x-coordinate). On the* **right**, *a scatter plot of height (y-coordinate) against weight (x-coordinate). I've put these plots next to one another so you don't have to mentally rotate (which is what you should usually do).*

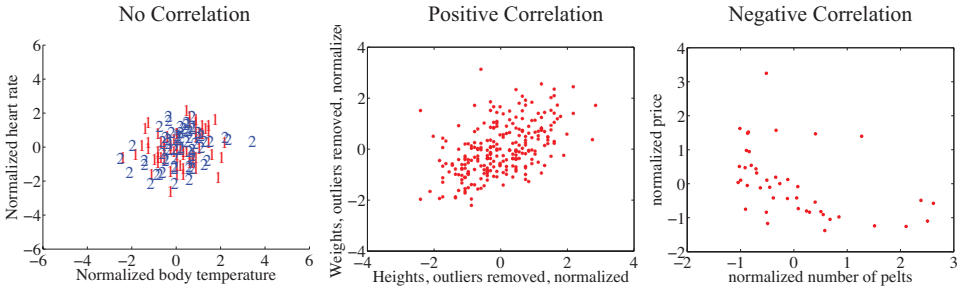

FIGURE 4.16: *The three kinds of scatter plot are less clean for real data than for our idealized examples. Here I used the body temperature vs heart rate data for the zero correlation; the height-weight data for positive correlation; and the lynx data for negative correlation. The pictures aren't idealized — real data tends to be messy — but you can still see the basic structures.*

The simplest, and most important, relationship to look for in a scatter plot is this: when $\hat{x}$ increases, does $\hat{y}$ tend to increase, decrease, or stay the same? This is straightforward to spot in a normalized scatter plot, because each case produces a very clear shape on the scatter plot. Any relationship is called **correlation** (we will see later how to measure this), and the three cases are: positive correlation, which means that larger $\hat{x}$ values tend to appear with larger $\hat{y}$ values; zero correlation, which means no relationship; and negative correlation, which means that larger $\hat{x}$ values tend to appear with smaller $\hat{y}$ values. I have shown these cases together in one figure using a real data example (Figure 4.16), so you can compare the appearance of the plots.

Positive correlation occurs when larger $\hat{x}$ values tend to appear with larger $\hat{y}$ values. This means that data points with with small (i.e. negative with large magnitude) $\hat{x}$ values must have small $\hat{y}$ values, otherwise the mean of $\hat{x}$ (resp. $\hat{y}$) would be too big. In turn, this means that the scatter plot should look like a "smear" of data from the bottom left of the graph to the top right. The smear might be broad or narrow, depending on some details we'll discuss below. Figure 4.12 shows normalized scatter plots of weight against height, and of body temperature against heart rate. In the weight-height plot, you can clearly see that individuals who are higher tend to weigh more. The important word here is "tend" — taller people could be lighter, but mostly they tend not to be. Notice, also, that I did NOT say that they weighed more *because* they were taller, but only that they tend to be heavier.

Zero correlation occurs when there is no relationship. This produces a characteristic shape in a scatter plot, but it takes a moment to understand why. If there really is no relationship, then knowing $\hat{x}$ will tell you nothing about $\hat{y}$. All we know is that $\mathsf{mean}(\{\hat{y}\}) = 0$, and $\mathsf{var}(\{\hat{y}\}) = 1$. Our value of $\hat{y}$ should have this mean and this variance, but it doesn't depend on $\hat{x}$ in any way. This is enough information to predict what the plot will look like. We know that $\mathsf{mean}(\{\hat{x}\}) = 0$ and $\mathsf{var}(\{\hat{x}\}) = 1$; so there will be many data points with $\hat{x}$ value close to zero, and few with a much larger or much smaller $\hat{x}$ value. The same applies to $\hat{y}$. Now

consider the data points in a strip of $\hat{x}$ values. If this strip is far away from the origin, there will be few data points in the strip, because there aren't many big $\hat{x}$ values. If there is no relationship, we don't expect to see large or small $\hat{y}$ values in this strip, because there are few data points in the strip and because large or small $\hat{y}$ values are uncommon — we see them only if there are many data points, and then seldom. So for a strip with $\hat{x}$ close to zero, we might see large $\hat{y}$ values; but for one that is far away, we expect to see small $\hat{y}$ values. We should see a blob, centered at the origin. In the temperature-heart rate plot of figure 4.13, it looks as though nothing of much significance is happening. The average heart rate seems to be about the same for people who run warm or who run cool. There is probably not much relationship here.

Negative correlation occurs when larger $\hat{x}$ values tend to appear with smaller $\hat{y}$ values. This means that data points with with small $\hat{x}$ values must have large $\hat{y}$ values, otherwise the mean of $\hat{x}$ (resp. $\hat{y}$) would be too big. In turn, this means that the scatter plot should look like a "smear" of data from the top left of the graph to the bottom right. The smear might be broad or narrow, depending on some details we'll discuss below. Figure 4.14 shows a normalized scatter plot of the lynx pelt-price data, where I have excluded the data from 1901 on. I did so because there seemed to be some other effect operating to drive prices up, which was inconsistent with the rest of the series. This plot suggests that when there were more pelts, prices were lower, as one would expect.

Notice that leaving out data, as I did here, should be done with care. If you exclude every data point that might disagree with your hypothesis, you may miss the fact that you are wrong. Leaving out data is an essential component of many kinds of fraud. You should always reveal whether you have excluded data, and why, to allow the reader to judge the evidence.

The correlation is not affected by which variable is plotted on the x-axis and which is plotted on the y-axis. Figure 4.15 compares a plot of height against weight to one of weight against height. Usually, one just does this by rotating the page, or by imagining the new picture. The left plot tells you that data points with higher height value tend to have higher weight value; the right plot tells you that data points with higher weight value tend to have higher height value — i.e. the plots tell you the same thing. It doesn't really matter which one you look at. Again, the important word is "tend" — the plot doesn't tell you anything about *why*, it just tells you that when one variable is larger the other tends to be, too.

4.2.1 The Correlation Coefficient

Consider a normalized data set of N two-dimensional vectors. We can write the i'th data point *in standard coordinates* $(\hat{x}_i, \hat{y}_i)$. We already know many important summaries of this data, because it is in standard coordinates. We have $\mathsf{mean}\left(\{\hat{x}\}\right) = 0$; $\mathsf{mean}\left(\{\hat{y}\}\right) = 0$; $\mathsf{std}\left(\hat{x}\right) = 1$; and $\mathsf{std}\left(\hat{y}\right) = 1$. Each of these summaries is itself the mean of some monomial. So $\mathsf{std}\left(\hat{x}\right)^2 = \mathsf{mean}\left(\left\{\hat{x}^2\right\}\right) = 1$; $\mathsf{std}\left(\hat{y}\right)^2 = \mathsf{mean}\left(\left\{\hat{y}^2\right\}\right)$ (the other two are easy). We can rewrite this information in terms of means of monomials, giving $\mathsf{mean}\left(\{\hat{x}\}\right) = 0$; $\mathsf{mean}\left(\{\hat{y}\}\right) = 0$; $\mathsf{mean}\left(\left\{\hat{x}^2\right\}\right) = 1$; and $\mathsf{mean}\left(\left\{\hat{y}^2\right\}\right) = 1$. There is one monomial missing here, which is $\hat{x}\hat{y}$.

The term $\mathsf{mean}\left(\{\hat{x}\hat{y}\}\right)$ captures correlation between x and y. The term is

known as the **correlation coefficient** or **correlation**.

Definition: 4.1 *Correlation coefficient*

Assume we have N data items which are 2-vectors $(x_1, y_1), \ldots, (x_N, y_N)$, where $N > 1$. These could be obtained, for example, by extracting components from larger vectors. We compute the correlation coefficient by first normalizing the x and y coordinates to obtain $\hat{x}_i = \frac{(x_i - \mathsf{mean}(\{x\}))}{\mathsf{std}(x)}$, $\hat{y}_i = \frac{(y_i - \mathsf{mean}(\{y\}))}{\mathsf{std}(y)}$. The correlation coefficient is the mean value of $\hat{x}\hat{y}$, and can be computed as:

$$\mathsf{corr}\left(\{(x, y)\}\right) = \frac{\sum_i \hat{x}_i \hat{y}_i}{N}$$

Correlation is a measure of our ability to predict one value from another. The correlation coefficient takes values between -1 and 1 (we'll prove this below). If the correlation coefficient is close to 1, then we are likely to predict very well. Small correlation coefficients (under about 0.5, say, but this rather depends on what you are trying to achieve) tend not to be all that interesting, because (as we shall see) they result in rather poor predictions. Figure 4.17 gives a set of scatter plots of different real data sets with different correlation coefficients. These all come from data set of age-height-weight, which you can find at `http://www2.stetson.edu/~jrasp/data.htm` (look for bodyfat.xls). In each case, two outliers have been removed. Age and height are hardly correlated, as you can see from the figure. Younger people do tend to be slightly taller, and so the correlation coefficient is -0.25. You should interpret this as a small correlation. However, the variable called "adiposity" (which isn't defined, but is presumably some measure of the amount of fatty tissue) is quite strongly correlated with weight, with a correlation coefficient is 0.86. Average tissue density is quite strongly negatively correlated with adiposity, because muscle is much denser than fat, so these variables are negatively correlated — we expect high density to appear with low adiposity, and vice versa. The correlation coefficient is -0.86. Finally, density is very strongly correlated with body weight. The correlation coefficient is -0.98.

It's not always convenient or a good idea to produce scatter plots in standard coordinates (among other things, doing so hides the units of the data, which can be a nuisance). Fortunately, scaling or translating data does not change the value of the correlation coefficient (though it can change the sign if one scale is negative). This means that it's worth being able to spot correlation in a scatter plot that isn't in standard coordinates (even though correlation is always *defined* in standard coordinates). Figure 4.18 shows different correlated datasets plotted in their original units. These data sets are the same as those used in figure 4.17

Properties of the Correlation Coefficient

You should memorize the following properties of the correlation coefficient:

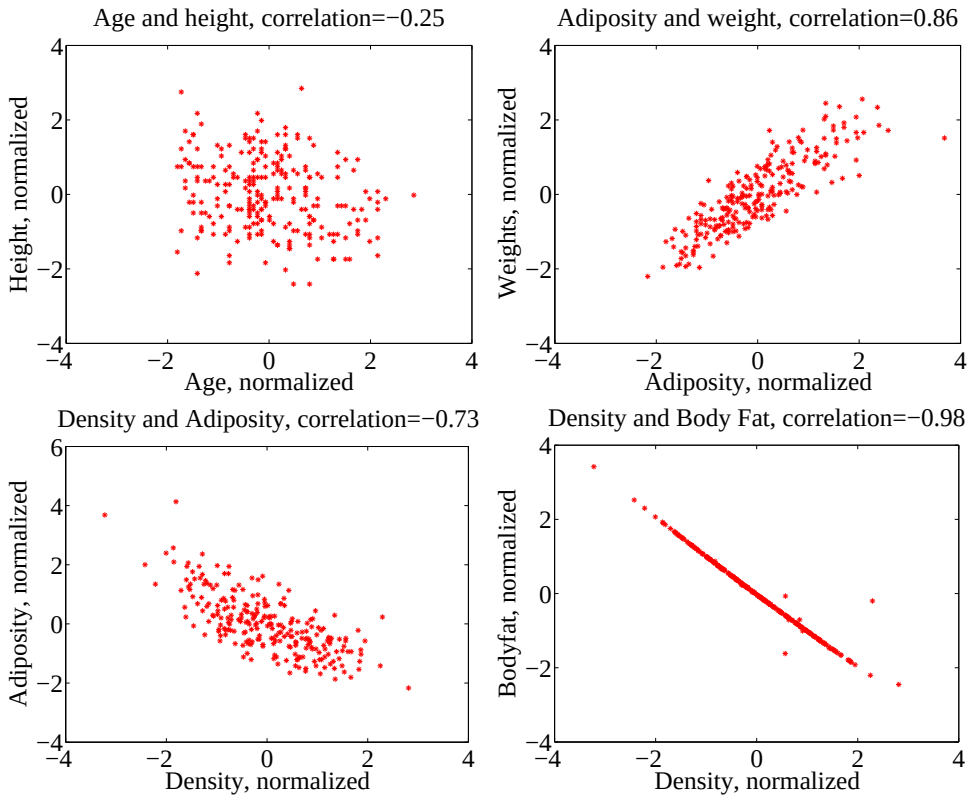

FIGURE 4.17: *Scatter plots for various pairs of variables for the age-height-weight dataset from* `http://www2.stetson.edu/~jrasp/data.htm`; *bodyfat.xls. In each case, two outliers have been removed, and the plots are in standard coordinates (compare to figure 4.18, which shows these data sets plotted in their original units). The legend names the variables.*

- The correlation coefficient is symmetric (it doesn't depend on the order of its arguments), so
$$\mathsf{corr}\left(\{(x,y)\}\right) = \mathsf{corr}\left(\{(y,x)\}\right)$$
- The value of the correlation coefficient is not changed by translating the data. Scaling the data can change the sign, but not the absolute value. For constants $a \neq 0$, b, $c \neq 0$, d we have
$$\mathsf{corr}\left(\{(ax+b,cx+d)\}\right) = \text{sign}(ab)\mathsf{corr}\left(\{(x,y)\}\right)$$
- If $\hat{y}$ tends to be large (resp. small) for large (resp. small) values of $\hat{x}$, then the correlation coefficient will be positive.
- If $\hat{y}$ tends to be small (resp. large) for large (resp. small) values of $\hat{x}$, then the correlation coefficient will be negative.

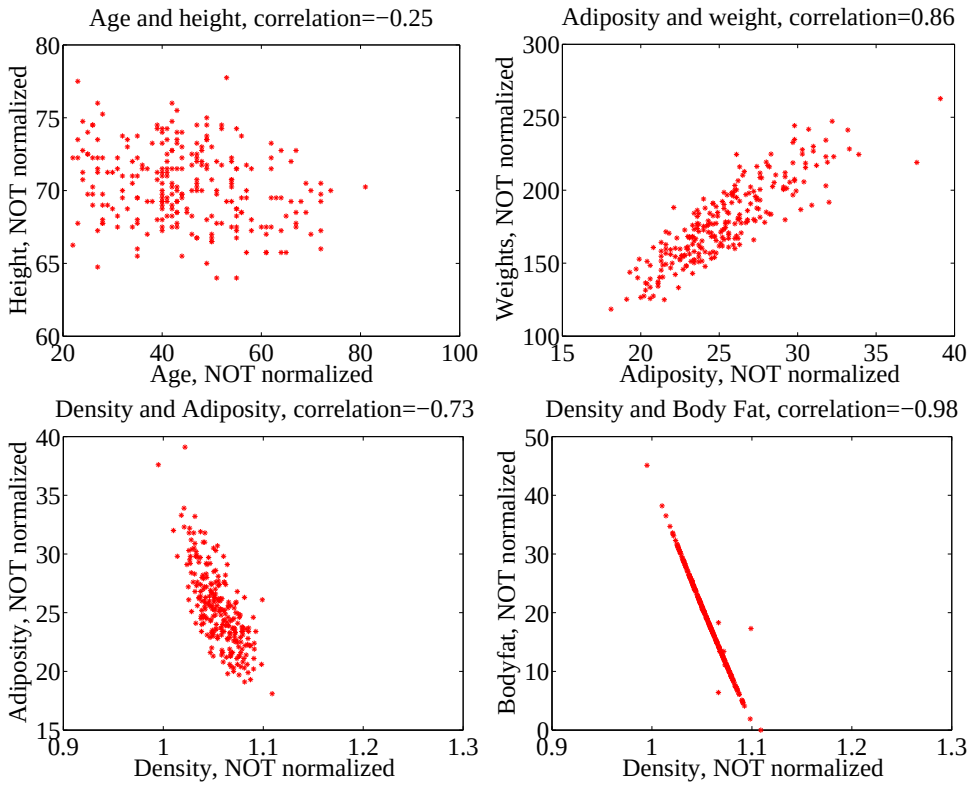

FIGURE 4.18: *Scatter plots for various pairs of variables for the age-height-weight dataset from* `http://www2.stetson.edu/~jrasp/data.htm`; *bodyfat.xls. In each case, two outliers have been removed, and the plots are* NOT *in standard coordinates (compare to figure 4.17, which shows these data sets plotted in normalized coordinates). The legend names the variables.*

- If $\hat{y}$ doesn't depend on $\hat{x}$, then the correlation coefficient is zero (or close to zero).
- The largest possible value is 1, which happens when $\hat{x} = \hat{y}$.
- The smallest possible value is -1, which happens when $\hat{x} = -\hat{y}$.

The first property is easy, and we relegate that to the exercises. One way to see that the correlation coefficient isn't changed by translation or scale is to notice that it is defined in standard coordinates, and scaling or translating data doesn't change those. Another way to see this is to scale and translate data, then write out the equations; notice that taking standard coordinates removes the effects of the scale and translation. In each case, notice that if the scale is negative, the sign of the correlation coefficient changes.

The property that, if $\hat{y}$ tends to be large (resp. small) for large (resp. small) values of $\hat{x}$, then the correlation coefficient will be positive, doesn't really admit

a formal statement. But it's relatively straightforward to see what's going on. Because $\mathsf{mean}\left(\{\hat{x}\}\right) = 0$, small values of $\mathsf{mean}\left(\{\hat{x}\}\right)$ must be negative and large values must be positive. But $\mathsf{corr}\left(\{(x,y)\}\right) = \frac{\sum_i \hat{x}_i \hat{y}_i}{N}$; and for this sum to be positive, it should contain mostly positive terms. It can contain few or no hugely positive (or hugely negative) terms, because $\mathsf{std}\left(\hat{x}\right) = \mathsf{std}\left(\hat{y}\right) = 1$ so there aren't many large (or small) numbers. For the sum to contain mostly positive terms, then the sign of $\hat{x}_i$ should be the same as the sign $\hat{y}_i$ for most data items. Small changes to this argument work to show that if if $\hat{y}$ tends to be small (resp. large) for large (resp. small) values of $\hat{x}$, then the correlation coefficient will be negative.

Showing that no relationship means zero correlation requires slightly more work. Divide the scatter plot of the dataset up into thin vertical strips. There are S strips. Each strip is narrow, so the $\hat{x}$ value does not change much for the data points in a particular strip. For the s'th strip, write $N(s)$ for the number of data points in the strip, $\hat{x}(s)$ for the $\hat{x}$ value at the center of the strip, and $\overline{\hat{y}}(s)$ for the mean of the $\hat{y}$ values within that strip. Now the strips are narrow, so we can approximate all data points within a strip as having the same value of $\hat{x}$. This yields

$$\mathsf{mean}\left(\{\hat{x}\hat{y}\}\right) \approx \frac{1}{S} \sum_{s \in \text{strips}} \hat{x}(s) \left[N(s)\overline{\hat{y}}(s)\right]$$

(where you could replace $\approx$ with $=$ if the strips were narrow enough). Now assume that $\hat{y}(s)$ does not change from strip to strip, meaning that there is no relationship between $\hat{x}$ and $\hat{y}$ in this dataset (so the picture is like the left hand side in figure 4.16). Then each value of $\overline{\hat{y}}(s)$ is the same — we write $\overline{\hat{y}}$ — and we can rearrange to get

$$\mathsf{mean}\left(\{\hat{x}\hat{y}\}\right) \approx \overline{\hat{y}} \frac{1}{S} \sum_{s \in \text{strips}} \hat{x}(s).$$

Now notice that

$$0 = \mathsf{mean}\left(\{\hat{y}\}\right) \approx \frac{1}{S} \sum_{s \in \text{strips}} N(s)\overline{\hat{y}}(s)$$

(where again you could replace $\approx$ with $=$ if the strips were narrow enough). This means that if every strip has the same value of $\overline{\hat{y}}(s)$, then that value must be zero. In turn, if there is no relationship between $\hat{x}$ and $\hat{y}$, we must have $\mathsf{mean}\left(\{\hat{x}\hat{y}\}\right) = 0$.

Proposition:

$$-1 \leq \mathsf{corr}\left(\{(x,y)\}\right) \leq 1$$

Proof: Writing $\hat{x}$, $\hat{y}$ for the normalized coefficients, we have

$$\mathsf{corr}\left(\{(x,y)\}\right) = \frac{\sum_i \hat{x}_i \hat{y}_i}{N}$$

and you can think of the value as the inner product of two vectors. We write

$$\mathbf{x} = \frac{1}{\sqrt{N}}\left[\hat{x}_1, \hat{x}_2, \ldots \hat{x}_N\right] \text{ and } \mathbf{y} = \frac{1}{\sqrt{N}}\left[\hat{y}_1, \hat{y}_2, \ldots \hat{y}_N\right]$$

and we have $\mathsf{corr}\left(\{(x,y)\}\right) = \mathbf{x}^T\mathbf{y}$. Notice $\mathbf{x}^T\mathbf{x} = \mathsf{std}\left(x\right)^2 = 1$, and similarly for $\mathbf{y}$. But the inner product of two vectors is at its maximum when the two vectors are the same, and this maximum is 1. This argument is also sufficient to show that smallest possible value of the correlation is -1, and this occurs when $\hat{x}_i = -\hat{y}_i$ for all i.

Property 4.1: The largest possible value of the correlation is 1, and this occurs when $\hat{x}_i = \hat{y}_i$ for all i. The smallest possible value of the correlation is -1, and this occurs when $\hat{x}_i = -\hat{y}_i$ for all i.

4.2.2 Using Correlation to Predict

Assume we have N data items which are 2-vectors $(x_1, y_1), \ldots, (x_N, y_N)$, where $N > 1$. These could be obtained, for example, by extracting components from larger vectors. As usual, we will write $\hat{x}_i$ for x_i in normalized coordinates, and so on. Now assume that we know the correlation coefficient is r (this is an important, traditional notation). What does this mean?

One (very useful) interpretation is in terms of prediction. Assume we have a data point $(x_0, ?)$ where we know the x-coordinate, but not the y-coordinate. We can use the correlation coefficient to predict the y-coordinate. First, we transform to standard coordinates. Now we must obtain the best $\hat{y}_0$ value to predict, using the $\hat{x}_0$ value we have.

We want to construct a prediction function which gives a prediction for any value of $\hat{x}$. This predictor should behave as well as possible on our existing data. For each of the $(\hat{x}_i, \hat{y}_i)$ pairs in our data set, the predictor should take $\hat{x}_i$ and produce a result as close to $\hat{y}_i$ as possible. We can choose the predictor by looking at the errors it makes at each data point.

We write $\hat{y}_i^p$ for the value of $\hat{y}_i$ predicted at $\hat{x}_i$. The simplest form of predictor is linear. If we predict using a linear function, then we have, for some unknown a, b, that $\hat{y}_i^p = a\hat{x}_i + b$. Now think about $u_i = \hat{y}_i - \hat{y}_i^p$, which is the error in our prediction. We would like to have $\mathsf{mean}\left(\{u\}\right) = 0$ (otherwise, we could reduce the

error of the prediction just by subtracting a constant).

$$\begin{aligned}
\mathsf{mean}\left(\{u\}\right) &= \mathsf{mean}\left(\{\hat{y} - \hat{y}^p\}\right) \\
&= \mathsf{mean}\left(\{\hat{y}\}\right) - \mathsf{mean}\left(\{a\hat{x}_i + b\}\right) \\
&= \mathsf{mean}\left(\{\hat{y}\}\right) - a\mathsf{mean}\left(\{\hat{x}\}\right) + b \\
&= 0 - a0 + b \\
&= 0.
\end{aligned}$$

This means that we must have $b = 0$.

To estimate a, we need to think about $\mathsf{var}\left(\{u\}\right)$. We should like $\mathsf{var}\left(\{u\}\right)$ to be as small as possible, so that the errors are as close to zero as possible (remember, small variance means small standard deviation which means the data is close to the mean). We have

$$\begin{aligned}
\mathsf{var}\left(\{u\}\right) &= \mathsf{var}\left(\{\hat{y} - \hat{y}^p\}\right) \\
&= \mathsf{mean}\left(\left\{(\hat{y} - a\hat{x})^2\right\}\right) \quad \text{because } \mathsf{mean}\left(\{u\}\right) = 0 \\
&= \mathsf{mean}\left(\left\{(\hat{y})^2 - 2a\hat{x}\hat{y} + a^2(\hat{x})^2\right\}\right) \\
&= \mathsf{mean}\left(\left\{(\hat{y})^2\right\}\right) - 2a\mathsf{mean}\left(\{\hat{x}\hat{y}\}\right) + a^2\mathsf{mean}\left(\left\{(\hat{x})^2\right\}\right) \\
&= 1 - 2ar + a^2,
\end{aligned}$$

which we want to minimize by choice of a. At the minimum, we must have

$$\frac{d\mathsf{var}\left(\{u_i\}\right)}{da} = 0 = -2r + 2a$$

so that $a = r$ and the correct prediction is

$$\hat{y}_0^p = r\hat{x}_0$$

You can use a version of this argument to establish that if we have $(?, \hat{y}_0)$, then the best prediction for $\hat{x}_0$ (*which is in standard coordinates*) is $r\hat{y}_0$. It is important to notice that the coefficient of $\hat{y}_i$ is NOT $1/r$; you should work this example, which appears in the exercises. We now have a prediction procedure, outlined below.

Procedure: 4.1 *Predicting a value using correlation*

Assume we have N data items which are 2-vectors $(x_1, y_1), \ldots, (x_N, y_N)$, where $N > 1$. These could be obtained, for example, by extracting components from larger vectors. Assume we have an x value x_0 for which we want to give the best prediction of a y value, based on this data. The following procedure will produce a prediction:

- Transform the data set into standard coordinates, to get

$$\begin{aligned}\hat{x}_i &= \frac{1}{\mathsf{std}\,(x)}(x_i - \mathsf{mean}\,(\{x\})) \\ \hat{y}_i &= \frac{1}{\mathsf{std}\,(y)}(y_i - \mathsf{mean}\,(\{y\})) \\ \hat{x}_0 &= \frac{1}{\mathsf{std}\,(x)}(x_0 - \mathsf{mean}\,(\{x\})).\end{aligned}$$

- Compute the correlation

$$r = \mathsf{corr}\,(\{(x, y)\}) = \mathsf{mean}\,(\{\hat{x}\hat{y}\}).$$

- Predict $\hat{y}_0 = r\hat{x}_0$.
- Transform this prediction into the original coordinate system, to get

$$y_0 = \mathsf{std}\,(y) r \hat{x}_0 + \mathsf{mean}\,(\{y\})$$

Now assume we have a y value y_0, for which we want to give the best prediction of an x value, based on this data. The following procedure will produce a prediction:

- Transform the data set into standard coordinates.
- Compute the correlation.
- Predict $\hat{x}_0 = r\hat{y}_0$.
- Transform this prediction into the original coordinate system, to get

$$x_0 = \mathsf{std}\,(x) r \hat{y}_0 + \mathsf{mean}\,(\{x\})$$

There is another way of thinking about this prediction procedure, which is often helpful. Assume we need to predict a value for x_0. In normalized coordinates, our prediction is $\hat{y}^p = r\hat{x}_0$; if we revert back to the original coordinate system, the

prediction becomes

$$\frac{(y^p - \mathsf{mean}\,(\{y\}))}{\mathsf{std}\,(y)} = r(\frac{(x_0 - \mathsf{mean}\,(\{x\}))}{\mathsf{std}\,(x)}).$$

This gives a really useful rule of thumb, which I have broken out in the box below.

Procedure: 4.2 *Predicting a value using correlation: Rule of thumb - 1*

If x_0 is k standard deviations from the mean of x, then the predicted value of y will be rk standard deviations away from the mean of y, and the sign of r tells whether y increases or decreases.

An even more compact version of the rule of thumb is in the following box.

Procedure: 4.3 *Predicting a value using correlation: Rule of thumb - 2*

The predicted value of y goes up by r standard deviations when the value of x goes up by one standard deviation.

We can compute the average root mean square error that this prediction procedure will make. The square of this error must be

$$\begin{aligned}\mathsf{mean}\left(\left\{u^2\right\}\right) &= \mathsf{mean}\left(\left\{y^2\right\}\right) - 2r\mathsf{mean}\left(\{xy\}\right) + r^2\mathsf{mean}\left(\left\{x^2\right\}\right) \\ &= 1 - 2r^2 + r^2 \\ &= 1 - r^2\end{aligned}$$

so the root mean square error will be $\sqrt{1-r^2}$. This is yet another intepretation of correlation; if x and y have correlation close to one, then predictions could have very small root mean square error, and so might be very accurate. In this case, knowing one variable is about as good as knowing the other. If they have correlation close to zero, then the root mean square error in a prediction might be as large as the root mean square error in $\hat{y}$ — which means the prediction is nearly a pure guess.

The prediction argument means that we can spot correlations for data in other kinds of plots — one doesn't have to make a scatter plot. For example, if we were to observe a child's height from birth to their 10'th year (you can often find these observations in ballpen strokes, on kitchen walls), we could plot height as a function of year. If we also had their weight (less easily found), we could plot weight as a function of year, too. The prediction argument above say that, if you can predict the weight from the height (or vice versa) then they're correlated. One way to spot this is to look and see if one curve goes up when the other does (or goes down when the other goes up). You can see this effect in figure 4.5, where (before 19h00), prices go down when the number of pelts goes up, and vice versa. These two variables are negatively correlated.

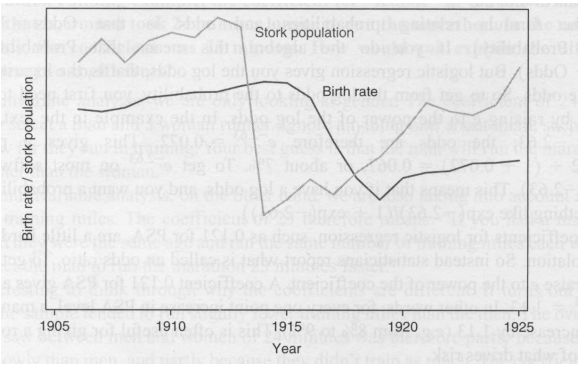

FIGURE 4.19: *This figure, from Vickers (ibid, p184) shows a plot of the stork population as a function of time, and the human birth rate as a function of time, for some years in Germany. The correlation is fairly clear; but this does not mean that reducing the number of storks means there are fewer able to bring babies. Instead, this is the impact of the first world war — a hidden or latent variable.*

4.2.3 Confusion caused by correlation

There is one very rich source of potential (often hilarious) mistakes in correlation. When two variables are correlated, they change together. If the correlation is positive, that means that, in typical data, if one is large then the other is large, and if one is small the other is small. In turn, this means that one can make a reasonable prediction of one from the other. However, correlation DOES NOT mean that changing one variable causes the other to change (sometimes known as causation).

Two variables in a dataset could be correlated for a variety of reasons. One important reason is pure accident. If you look at enough pairs of variables, you may well find a pair that appears to be correlated just because you have a small set of observations. Imagine, for example, you have a dataset consisting of only two vectors — there is a pretty good chance that there is some correlation between the coefficients. Such accidents can occur in large datasets, particularly if the dimensions are high.

Another reason variables could be correlated is that there is some causal relationship — for example, pressing the accelerator tends to make the car go faster, and so there will be some correlation between accelerator position and car acceleration. As another example, adding fertilizer does tend to make a plant grow bigger. Imagine you record the amount of fertilizer you add to each pot, and the size of the resulting potplant. There should be some correlation.

Yet another reason variables could be correlated is that there is some other background variable — often called a **latent variable** — linked causally to each of the observed variables. For example, in children (as Freedman, Pisani and Purves note in their excellent *Statistics*), shoe size is correlated with reading skills. This DOES NOT mean that making your feet grow will make you read faster, or that you can make your feet shrink by forgetting how to read. The real issue here is

the age of the child. Young children tend to have small feet, and tend to have weaker reading skills (because they've had less practice). Older children tend to have larger feet, and tend to have stronger reading skills (because they've had more practice). You can make a reasonable prediction of reading skills from foot size, because they're correlated, even though there is no direct connection.

This kind of effect can mask correlations, too. Imagine you want to study the effect of fertilizer on potplants. You collect a set of pots, put one plant in each, and add different amounts of fertilizer. After some time, you record the size of each plant. You expect to see correlation between fertilizer amount and plant size. But you might not if you had used a different species of plant in each pot. Different species of plant can react quite differently to the same fertilizer (some plants just die if over-fertilized), so the species could act as a latent variable. With an unlucky choice of the different species, you might even conclude that there was a negative correlation between fertilizer and plant size. This example illustrates why you need to take great care in setting up experiments and interpreting their results.

This sort of thing happens often, and it's an effect you should look for. Another nice example comes from Vickers (ibid). The graph, shown in Figure 4.19, shows a plot of (a) a dataset of the stork population in Europe over a period of years and (b) a dataset of the birth rate over those years. This isn't a scatter plot; instead, the data has been plotted on a graph. You can see by eye that these two datasets are quite strongly correlated . Even more disturbing, the stork population dropped somewhat before the birth rate dropped. Is this evidence that storks brought babies in Europe during those years? No (the usual arrangement seems to have applied). For a more sensible explanation, look at the dates. The war disturbed both stork and human breeding arrangements. Storks were disturbed immediately by bombs, etc., and the human birth rate dropped because men died at the front.

4.3 STERILE MALES IN WILD HORSE HERDS

Large herds of wild horses are (apparently) a nuisance, but keeping down numbers by simply shooting surplus animals would provoke outrage. One strategy that has been adopted is to sterilize males in the herd; if a herd contains sufficient sterile males, fewer foals should result. But catching stallions, sterilizing them, and reinserting them into a herd is a performance — does this strategy work?

We can get some insight by plotting data. At `http://lib.stat.cmu.edu/DASL/Datafiles/WildHorses.html`, you can find a dataset covering herd management in wild horses. I have plotted part of this dataset in figure 4.20. In this dataset, there are counts of all horses, sterile males, and foals made on each of a small number of days in 1986, 1987, and 1988 for each of two herds. I extracted data for one herd. I have plotted this data as a function of the count of days since the first data point, because this makes it clear that some measurements were taken at about the same time, but there are big gaps in the measurements. In this plot, the data points are shown with a marker. Joining them leads to a confusing plot because the data points vary quite strongly. However, notice that the size of the herd drifts down slowly (you could hold a ruler against the plot to see the trend), as does the number of foals, when there is a (roughly) constant number of sterile

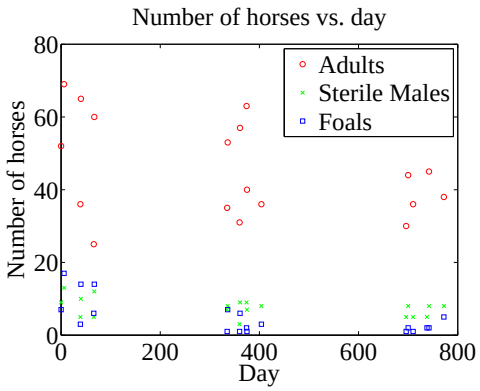

FIGURE 4.20: *A plot of the number of adult horses, sterile males, and foals in horse herds over a period of three years. The plot suggests that introducing sterile males might cause the number of foals to go down. Data from* `http://lib.stat.cmu.edu/DASL/Datafiles/WildHorses.html`.

males.

Does sterilizing males result in fewer foals? This is likely hard to answer for this dataset, but we could ask whether herds with more sterile males have fewer foals. A scatter plot is a natural tool to attack this question. However, the scatter plots of figure 4.21 suggest, rather surprisingly, that when there are more sterile males there are more adults (and vice versa), and when there are more sterile males there are more foals (and vice versa). This is borne out by a correlation analysis. The correlation coefficient between foals and sterile males is 0.74, and the correlation coefficient between adults and sterile males is 0.68. You should find this very surprising — how do the horses know how many sterile males there are in the herd? You might think that this is an effect of scaling the plot, but there is a scatter plot in normalized coordinates in figure 4.21 that is entirely consistent with the conclusions suggested by the unnormalized plot. What is going on here?

The answer is revealed by the scatter plots of figure 4.22. Here, rather than plotting a '*' at each data point, I have plotted the day number of the observation. This is in days from the first observation. You can see that the whole herd is shrinking — observations where there are many adults (resp. sterile adults, foals) occur with small day numbers, and observations where there are few have large day numbers. Because the whole herd is shrinking, it is true that when there are more adults and more sterile males, there are also more foals. Alternatively, you can see the plots of figure 4.20 as a scatter plot of herd size (resp. number of foals, number of sterile males) against day number. Then it becomes clear that the whole herd is shrinking, as is the size of each group. To drive this point home, we can look at the correlation coefficient between adults and days (-0.24), between sterile adults and days (-0.37), and between foals and days (-0.61). We can use the rule of thumb in box 3 to interpret this. This means that every 282 days, the herd loses about three adults; about one sterile adult; and about three foals. For the herd to have a stable size, it needs to gain by birth as many foals as it loses both to growing up

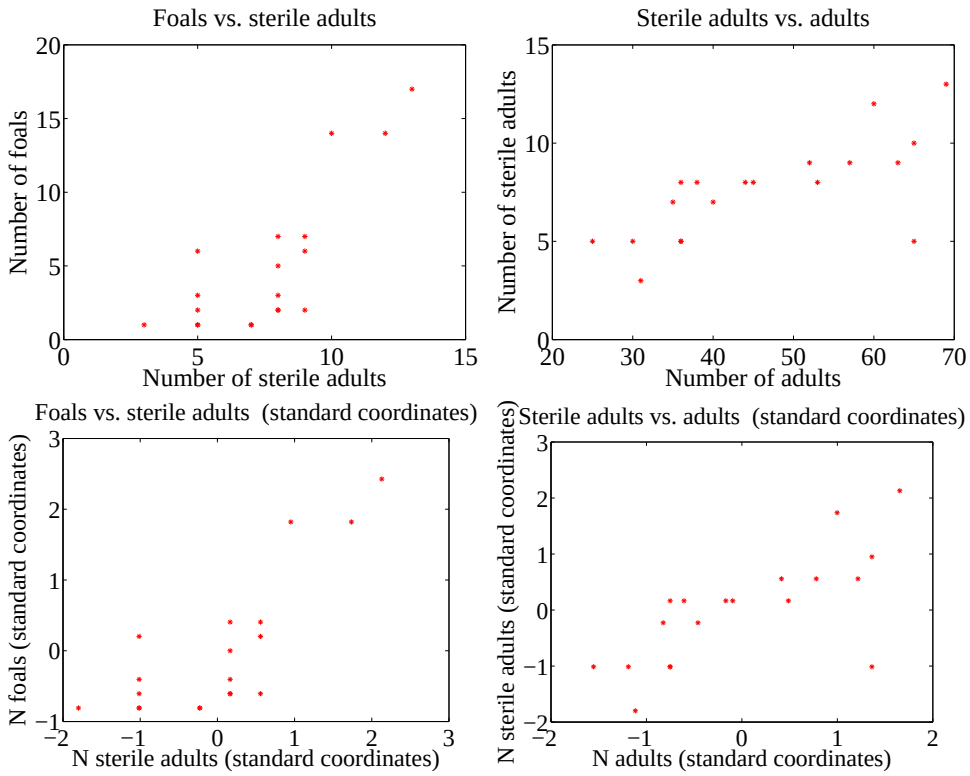

FIGURE 4.21: *Scatter plots of the number of sterile males in a horse herd against the number of adults, and the number of foals against the number of sterile males, from data of* `http://lib.stat.cmu.edu/DASL/Datafiles/WildHorses.html`. **Top:** *unnormalized;* **bottom:** *standard coordinates.*

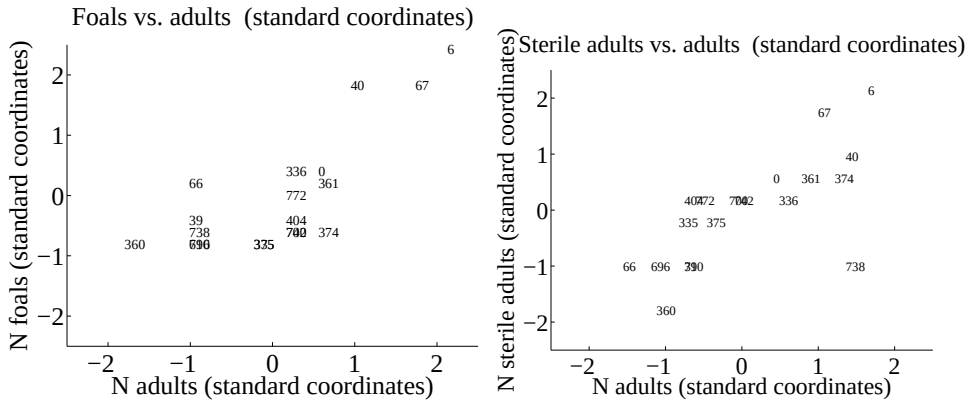

FIGURE 4.22:

and to death. If the herd is losing three foals every 282 days, then if they all grow up to replace the missing adults, the herd will be shrinking slightly (because it is losing four adults in this time); but if it loses foals to natural accidents, etc., then it is shrinking rather fast.

The message of this example is important. To understand a simple dataset, you might need to plot it several ways. You should make a plot, look at it and ask what it says, and then try to use another type of plot to confirm or refute what you think might be going on.

4.4 WHAT YOU MUST REMEMBER

You should be able to:

- Plot a bar chart, a heat map, and a pie chart for a categorical dataset.
- Plot a dataset as a graph, making sensible choices about markers, lines and the like.
- Plot a scatter plot for a dataset.
- Plot a normalized scatter plot for a dataset.
- Interpret the scatter plot to tell the sign of the correlation between two variables, and estimate the size of the correlation coefficient.
- Compute a correlation coefficient.
- Interpret a correlation coefficient.
- Use correlation to make predictions.

You should remember:

- The definition and properties of the correlation coefficient.

PROBLEMS

4.1. Show that $\mathsf{corr}(\{(x, y)\}) = \mathsf{corr}(\{(y, x)\})$ by substituting into the definition.

4.2. Show that if $\hat{y}$ tends to be small (resp. large) for large (resp. small) values of $\hat{x}$, then the correlation coefficient will be negative.

4.3. We have a 2D dataset consisting of N pairs $(\hat{x}_i, \hat{y}_i)$ in normalized coordinates. This data has correlation coefficient r. We observe a new $\hat{y}$ value $\hat{y}_0$, and wish to predict the (unknown) x value. We will do so with a linear prediction, choosing a, b, to predict an $\hat{x}$ for any $\hat{y}$ using the rule $\hat{x}^p = a\hat{y}^p + b$. Write $u_i = \hat{x}_i - \hat{x}_i^p$ for the error that this rule makes on each data item.

(a) We require $\mathsf{mean}(\{u\}) = 0$. Show that this means that $b = 0$.

(b) We require that $\mathsf{var}(\{u\})$ is minimized. Show that this means that $a = r$.

(c) We now have a result that seems paradoxical — if I have $(\hat{x}_0, ?)$ I predict $(\hat{x}_0, r\hat{x}_0)$ and if I have $(?, y_0)$, I predict $(r\hat{y}_0, \hat{y}_0)$. Use figure 4.23 to explain why this is right. The important difference between the two lines is that lies (approximately) in the middle of each vertical span of data, and the other lies (approximately) in the middle of each horizontal span of data.

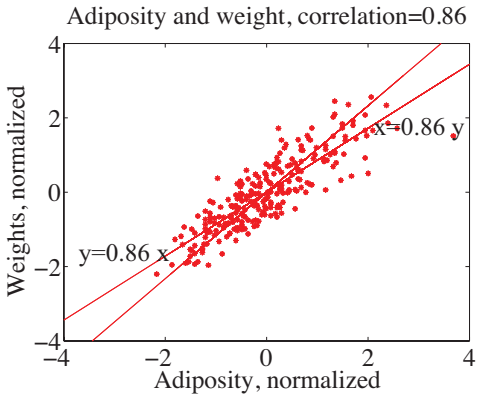

FIGURE 4.23: *This figure shows two lines, $y = 0.86x$ and $x = 0.86y$, superimposed on the normalized adiposity-weight scatter plot.*

4.4. I did the programming exercise about the earth temperature below. I looked at the years 1965-2012. Write $\{(y, T)\}$ for the dataset giving the temperature (T) of the earth in year y. I computed: $\mathsf{mean}\,(\{y\}) = 1988.5$, $\mathsf{std}\,(y) = 14$, $\mathsf{mean}\,(\{T\}) = 0.175$, $\mathsf{std}\,(T) = 0.231$ and $\mathsf{corr}\,(\{y\})T = 0.892$. What is the best prediction using this information for the temperature in mid 2014? in mid 2028? in mid 2042?

4.5. I did the programming exercise about the earth temperature below. It is straightforward to build a dataset $\{(T, n_t)\}$ where each entry contains the temperature of the earth (T) and the number of counties where FEMA declared tornadoes n_t (for each year, you look up T and n_t, and make a data item). I computed: $\mathsf{mean}\,(\{T\}) = 0.175$, $\mathsf{std}\,(T) = 0.231$, $\mathsf{mean}\,(\{n_t\}) = 31.6$, $\mathsf{std}\,(n_t) = 30.8$, and $\mathsf{corr}\,(\{T\})n_t = 0.471$. What is the best prediction using this information for the number of tornadoes if the global earth temperature is 0.5? 0.6? 0.7?

PROGRAMMING EXERCISES

4.6. At `http://lib.stat.cmu.edu/DASL/Datafiles/cigcancerdat.html`, you will find a dataset recording per capita cigarette sales and cancer deaths per 100K population for a variety of cancers, recorded for 43 states and the District of Columbia in 1960.

(a) Plot a scatter plot of lung cancer deaths against cigarette sales, using the two letter abbreviation for each state as a marker. You should see two fairly obvious outliers. The backstory at `http://lib.stat.cmu.edu/DASL/Stories/cigcancer.html` suggests that the unusual sales in Nevada are generated by tourism (tourists go home, and die there) and the unusual sales in DC are generated by commuting workers (who also die at home).

(b) What is the correlation coefficient between per capita cigarette sales and lung cancer deaths per 100K population? Compute this with, and without the outliers. What effect did the outliers have? Why?

(c) What is the correlation coefficient between per capita cigarette sales and bladder cancer deaths per 100K population? Compute this with, and

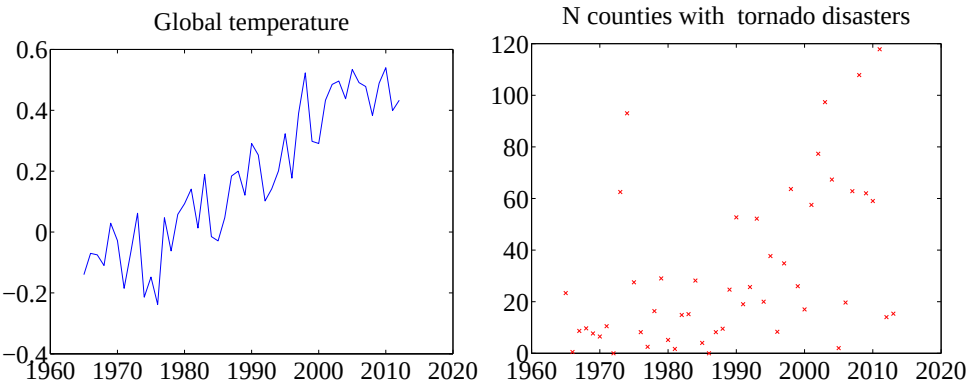

FIGURE 4.24: *Plots I prepared from* **left** *uea data on temperature and* **right** *FEMA data on tornadoes by county. These should help you tell if you're on the right track.*

without the outliers. What effect did the outliers have? Why?

(d) What is the correlation coefficient between per capita cigarette sales and kidney cancer deaths per 100K population? Compute this with, and without the outliers. What effect did the outliers have? Why?

(e) What is the correlation coefficient between per capita cigarette sales and leukemia deaths per 100K population? Compute this with, and without the outliers. What effect did the outliers have? Why?

(f) You should have computed a positive correlation between cigarette sales and lung cancer deaths. Does this mean that smoking causes lung cancer? Why?

(g) You should have computed a negative correlation between cigarette sales and leukemia deaths. Does this mean that smoking cures leukemia? Why?

4.7. At `http://www.cru.uea.ac.uk/cru/info/warming/gtc.csv`, you can find a dataset of global temperature by year. When I accessed this, the years spanned 1880-2012. I don't know what units the temperatures are measured in. Keep in mind that measuring the temperature of the earth has non-trivial difficulties (you can't just insert an enormous thermometer!), and if you look at `http://www.cru.uea.ac.uk/cru` and `http://www.cru.uea.ac.uk/cru/data/temperature/` you can see some discussion of the choices made to get these measurements. There are two kinds of data in this dataset, smoothed and unsmoothed. I used the unsmoothed data, which should be fine for our purposes. The government publishes a great deal of data at `http://data.gov`. From there, I found a dataset, published by the Federal Emergency Management Agency (FEMA), of all federally declared disasters (which I found at `http://www.fema.gov/media-library/assets/documents/28318?id=6292`). We would like to see whether weather related disasters are correlated to global temperature.

(a) The first step is preprocessing the data. The FEMA data has all sorts of information. From 1965 on, a disaster was declared per county (so you get one line in the data set for each county), but before 1965, it seems to have been by state. We divide the disasters into four types: TORNADO, FLOOD, STORM, HURRICANE. (FEMA seems to have a much richer

type system). We want to know how many counties declare a disaster of each type, in each year. This is a really rough estimate of the number of people affected by the disaster. If a disaster has two types (in some rows, you will see "SEVERE STORMS, HEAVY RAINS & FLOODING" we will allocate the credit evenly between the types (i.e. for this case we would count 1/2 for STORM and 1/2 for FLOOD). You should write code that will (a) read the dataset and then (b) compute a table with the count of the number of counties where a disaster of each type has occurred for each year. This takes a bit of work. Notice you only need to deal with two columns of the data (the date it was declared, and the type). Notice also that FEMA changed the way it represented dates somewhere through the first column (they added times), which can cause problems. You can tell the type of the disaster by just using a string match routine with the four keywords. Figure 4.24 shows the plot of temperature and of number of counties where FEMA declared a tornado disaster for this data.

(b) Plot a normalized scatter plot of the number of counties where FEMA declared the disaster against temperature, for each kind.

(c) For each kind of disaster, compute the correlation coefficient between the number of counties where FEMA declared the disaster and the year. For each kind of disaster, use this correlation coefficient to predict the number of disasters of this kind for 2013. Compare this to the true number, and explain what you see.

(d) For each kind of disaster, compute the correlation coefficient between the number of counties where FEMA declared the disaster and the global temperature. For each kind of disaster, use this correlation coefficient to predict the number of disasters of this kind when the earth reaches 0.6 temperature units and 0.7 temperature units (on the absolute temperature scale).

(e) Does this data show that warming of the earth causes weather disasters? Why?

(f) Does this data suggest that more people will be affected by disasters in the US in the future? Why?

(g) Does this data suggest that the earth will be warmer in the future? Why?

4.8. If you go to `https://github.com/TheUpshot/Military-Surplus-Gear`, you will find data on purchases of military weapons by US police departments. This data is organized by state and county. There's a fair amount of data here, and you'll need to do some data jockeying.

(a) Prepare a plot showing how much each Illinois county spent under this program.

(b) Now look up population levels in the counties. Prepare a plot showing how much each county spent *per capita*.

(c) Prepare a graphic illustrating what the overall most popular items were — i.e., those items counties bought the most of.

(d) Prepare a graphic illustrating on what items the most money was spent — for example, was more money spent on "RIFLE,5.56 MILLIMETER" or on "MINE RESISTANT VEHICLE"?

(e) Prepare a graphic illustrating the pattern of purchases across counties for the ten overall most popular items.

(f) Can you draw any interesting conclusions?

CHAPTER 5

Basic ideas in probability

We will perform experiments — which could be pretty much anything, from flipping a coin, to eating too much saturated fat, to smoking, to crossing the road without looking — and reason about the outcomes (mostly bad, for the examples I gave). But these outcomes are uncertain, and we need to weigh those uncertainties against one another. If I flip a coin, I could get heads or tails, and there's no reason to expect to see one more often than the other. If I eat too much saturated fat or smoke, I will very likely have problems, though I might not. If I cross the road without looking, I may be squashed by a truck or I may not. Our methods need also to account for information. If I look before I cross the road, I am much less likely to be squashed.

Probability is the machinery we use to predict what will happen in an experiment. Probability measures the tendency of events to occur frequently or seldom when we repeat experiments. Building this machinery involves a formal model of potential outcomes and sets of outcomes. Once we have this, a set of quite simple axioms allows us, at least in principle, to compute probabilities in most situations.

5.1 EXPERIMENTS, OUTCOMES, EVENTS, AND PROBABILITY

If we flip a fair coin many times, we expect it to come up heads about as often as it comes up tails. If we toss a fair die many times, we expect each number to come up about the same number of times. We are performing an experiment each time we flip the coin, and each time we toss the die. We can formalize this experiment by describing the set of **outcomes** that we expect from the experiment. Every run of the experiment produces one of the set of possible outcomes. There are never two or more outcomes, and there is never no outcome.

In the case of the coin, the set of outcomes is:

$$\{H, T\}.$$

In the case of the die, the set of outcomes is:

$$\{1, 2, 3, 4, 5, 6\}.$$

We are making a modelling choice by specifying the outcomes of the experiment. For example, we are assuming that the coin can only come up heads or tails (but doesn't stand on its edge; or fall between the floorboards; or land behind the bookcase; or whatever). We write the set of all outcomes Ω; this is usually known as the **sample space**.

Worked example 5.1 *Find the lady*

We have three playing cards. One is a queen; one is a king, and one is a knave. All are shown face down, and one is chosen at random and turned up. What is the set of outcomes?

Solution: Write Q for queen, K for king, N for knave; the outcomes are $\{Q, K, N\}$

Worked example 5.2 *Find the lady, twice*

We play Find the Lady twice, replacing the card we have chosen. What is the sample space?

Solution: We now have $\{QQ, QK, QN, KQ, KK, KN, NQ, NK, NN\}$

Worked example 5.3 *Children*

A couple decides to have children until either (a) they have both a boy and a girl or (b) they have three children. What is the set of outcomes?

Solution: Write B for boy, G for girl, and write them in birth order; we have $\{BG, GB, BBG, BBB, GGB, GGG\}$.

Worked example 5.4 *Monty Hall (sigh!)*

There are three boxes. There is a goat, a second goat, and a car. These are placed into the boxes at random. The goats are indistinguishable for our purposes; equivalently, we do not care about the difference between goats. What is the sample space?

Solution: Write G for goat, C for car. Then we have $\{CGG, GCG, GGC\}$.

Worked example 5.5 *Monty Hall, different goats (sigh!)*

There are three boxes. There is a goat, a second goat, and a car. These are placed into the boxes at random. One goat is male, the other female, and the distinction is important. What is the sample space?

Solution: Write M for male goat, F for female goat, C for car. Then we have $\{CFM, CMF, FCM, MCF, FMC, MFC\}$. Notice how the number of outcomes has increased, because we now care about the distinction between goats.

Worked example 5.6 *A poor choice of strategy for planning a family*

A couple decides to have children. As they know no mathematics, they decide to have children until a girl then a boy are born. What is the sample space? Does this strategy bound the number of children they could be planning to have?

Solution: Write B for boy, G for girl. The sample space looks like any string of B's and G's that (a) ends in GB and (b) does not contain any other GB. In regular expression notation, you can write such strings as B^*G^+B. There is a lower bound (two), but no upper bound. As a family planning strategy, this is unrealistic, but it serves to illustrate the point that sample spaces don't have to be finite to be tractable.

5.1.1 The Probability of an Outcome

We represent our model of how often a particular outcome will occur in a repeated experiment with a **probability**, a non-negative number. This number gives the relative frequency of the outcome of interest, when an experiment is repeated a very large number of times.

Assume an outcome A has probability P. Assume we repeat the experiment a very large number of times N, and assume that the coins, dice, whatever don't communicate with one another from experiment to experiment (or, equivalently, that experiments don't "know" about one another). Then, for about $N \times P$ of those experiments the outcome will occur. Furthermore, as N gets larger, the fraction where the outcome occurs will get closer to P. We write $\#(A)$ for the number of times outcome A occurs. We interpret P as

$$\lim_{N \to \infty} \frac{\#(A)}{N}.$$

We can draw two important conclusions immediately.

- For any outcome A, $0 \leq P(A) \leq 1$.
- $\sum_{A_i \in \Omega} P(A_i) = 1$.

Remember that every run of the experiment produces exactly one outcome. The probabilities add up to one because each experiment must have one of the outcomes in the sample space.

Worked example 5.7 *A biased coin*

Assume we have a coin where the probability of getting heads is $P(H) = \frac{1}{3}$, and so the probability of getting tails is $P(T) = \frac{2}{3}$. We flip this coin three million times. How many times do we see heads?

Solution: $P(H) = \frac{1}{3}$, so we expect this coin will come up heads in $\frac{1}{3}$ of experiments. This means that we will very likely see very close to a million heads.

Some problems can be handled by building a set of outcomes and reasoning about the probability of each outcome. This is particularly useful when the outcomes *must* have the same probability, which happens rather a lot.

Worked example 5.8 *Find the Lady*

Assume that the card that is chosen is chosen fairly — that is, each card is chosen with the same probability. What is the probability of turning up a Queen?

Solution: There are three outcomes, and each is chosen with the same probability, so the probability is 1/3.

Worked example 5.9 *Monty Hall, indistinguishable goats, again*

Each outcome has the same probability. We choose to open the first box. With what probability will we find a goat (any goat)?

Solution: There are three outcomes, each has the same probability, and two give a goat, so 2/3

Worked example 5.10 *Monty Hall, yet again*

Each outcome has the same probability. We choose to open the first box. With what probability will we find the car?

Solution: There are three places the car could be, each has the same probability, so 1/3

Worked example 5.11 *Monty Hall, with distinct goats, again*

Each outcome has the same probability. We choose to open the first box. With what probability will we find a female goat?

Solution: Using the reasoning of the previous example, but substituting "female goat" for "car", 1/3. The point of this example is that the sample space matters. If you care about the gender of the goat, then it's important to keep track of it; if you don't, it's a good idea to omit it from the sample space.

5.1.2 Events

Assume we run an experiment and get an outcome. We know what the outcome is (that's the whole point of a sample space). This means we can tell whether the outcome we get belongs to some particular known *set* of outcomes. We just look in the set and see if our outcome is there. This means that we should be able to predict the probability of a *set* of outcomes from any reasonable model of an experiment. For example, we might roll a die and ask what the probability of getting an even number is. As another example, we might flip a coin ten times, and ask what the probability of getting three heads is.

An **event** is a set of outcomes. The set of all outcomes, which we wrote Ω, must be an event. It is not a particularly interesting event, because we must have $P(\Omega) = 1$ (because we said that every run of an experiment produces one outcome, and that outcome must be in Ω). In principle, there could be no outcome, although this never happens. This means that the empty set, which we write $\emptyset$, is an event, and we have $P(\emptyset) = 0$. The space of events has a rich structure, which makes it possible to compute probabilities for other, more interesting, events.

Notation: Generally, we write sets like $\mathcal{A}$; in principle, you could confuse this notation with the matrix notation, but it's clear from context which is meant. We write $\mathcal{A} \cup \mathcal{B}$ for the union of two sets, $\mathcal{A} \cap \mathcal{B}$ for the intersection of two sets, and $\mathcal{A} - \mathcal{B}$ for the set theoretic difference (i.e. $\mathcal{A} - \mathcal{B} = \{x \in \mathcal{A} | x \notin \mathcal{B}\}$). We will write $\Omega - \mathcal{U}$ as $\mathcal{U}^c$; read "the complement of $\mathcal{U}$".

Events have three important properties that follow from their nature as sets of outcomes:

- If $\mathcal{U}$ and $\mathcal{V}$ are events — sets of outcomes — then so is $\mathcal{U} \cap \mathcal{V}$. You should interpret this as the event that we have an outcome that is in $\mathcal{U}$ and also in $\mathcal{V}$.
- If $\mathcal{U}$ and $\mathcal{V}$ are events, then $\mathcal{U} \cup \mathcal{V}$ is also an event. You should interpret this as the event that we have an outcome that is either in $\mathcal{U}$ or in $\mathcal{V}$ (or in both).
- If $\mathcal{U}$ is an event, then $\mathcal{U}^c = \Omega - \mathcal{U}$ is also an event. You should think of this as the event we get an outcome that is not in $\mathcal{U}$.

This means that the set of all possible events Σ has a very important structure.

- $\emptyset$ is in Σ.
- Ω is in Σ.
- If $\mathcal{U} \in \Sigma$ and $\mathcal{V} \in \Sigma$ then $\mathcal{U} \cup \mathcal{V} \in \Sigma$.
- If $\mathcal{U} \in \Sigma$ and $\mathcal{V} \in \Sigma$ then $\mathcal{U} \cap \mathcal{V} \in \Sigma$.
- If $\mathcal{U} \in \Sigma$ then $\mathcal{U}^c \in \Sigma$.

This means that the space of events can be quite big. For a single flip of a coin, the only possible space of events looks like:

$$\{\emptyset, \{H\}, \{T\}, \{H, T\}\}$$

Many experiments admit more than one space of events. For example, if we flip two coins, one natural event space is

$$\left\{ \begin{array}{l} \emptyset, \Omega, \\ \{HH\}, \{HT\}, \{TH\}, \{TT\}, \\ \{HH, HT\}, \{HH, TH\}, \{HH, TT\}, \{HT, TT\}, \{HT, TH\}, \{TT, HH\}, \\ \{HT, TH, TT\}, \{HH, TH, TT\}, \{HH, HT, TT\}, \{HH, HT, TH\} \end{array} \right\}$$

which can represent any possible event that can be built out of two coin flips. But this is not the only event space possible with these outcomes.

Worked example 5.12 *The structure of event spaces*

I flip two coins. Is the following collection of sets an event space?

$$\Sigma = \{\emptyset, \Omega, \{HH\}, \{TT, HT, TH\}\}$$

Solution: Yes, because: $\emptyset \in \Sigma$; $\Omega \in \Sigma$; if $\mathcal{A} \in \Sigma$, $\mathcal{A}^c \in \Sigma$; if $\mathcal{A} \in \Sigma$ and $\mathcal{B} \in \Sigma$, $\mathcal{A} \cup \mathcal{B} \in \Sigma$; and if $\mathcal{A} \in \Sigma$ and $\mathcal{B} \in \Sigma$, $\mathcal{A} \cap \mathcal{B} \in \Sigma$.

So, from example 12, we can have different consistent collections of events built on top of the same set of outcomes. This makes sense, because it allows us to reason about different kinds of result obtained with the same experimental equipment. You can interpret the event space in example 12 as encoding the events "two heads" and "anything other than two heads".

For a single throw of the die, the set of every possible event is

$$\left\{\begin{array}{llllll}
\emptyset, & \{1,2,3,4,5,6\}, & & & & \\
\{1\}, & \{2\}, & \{3\}, & \{4\}, & \{5\}, & \{6\}, \\
 & \{1,2\}, & \{1,3\}, & \{1,4\}, & \{1,5\}, & \{1,6\}, \\
 & & \{2,3\}, & \{2,4\}, & \{2,5\}, & \{2,6\}, \\
 & & & \{3,4\}, & \{3,5\}, & \{3,6\}, \\
 & & & & \{4,5\}, & \{4,6\}, \\
 & & & & & \{5,6\}, \\
 & \{1,2,3\}, & \{1,2,4\}, & \{1,2,5\}, & \{1,2,6\}, & \\
 & & \{1,3,4\}, & \{1,3,5\}, & \{1,3,6\}, & \\
 & & & \{1,4,5\}, & \{1,4,6\}, & \\
 & & & & \{1,5,6\}, & \\
 & & \{2,3,4\}, & \{2,3,5\}, & \{2,3,6\}, & \\
 & & \{2,4,5\}, & \{2,4,6\}, & \{2,5,6\}, & \\
 & & & \{3,4,5\}, & \{3,4,6\}, & \\
 & & & & \{3,5,6\}, & \\
 & & & & \{4,5,6\}, & \\
 & & \{1,2,3,4\}, & \{1,2,3,5\}, & \{1,2,3,6\}, & \\
 & & & \{1,3,4,5\}, & \{1,3,4,6\}, & \\
 & & & \{2,3,4,5\}, & \{2,3,4,6\}, & \\
 & & & & \{3,4,5,6\}, & \\
\{2,3,4,5,6\}, & \{1,3,4,5,6\}, & \{1,2,4,5,6\}, & \{1,2,3,5,6\}, & \{1,2,3,4,6\}, & \{1,2,3,4,5\}
\end{array}\right\}$$

(which gives some explanation as to why we don't usually write out the whole space of events). In fact, it is seldom necessary to explain which event space one is working with. We usually assume that the event space consists of all events that can be obtained with the outcomes (as in the event space shown for a die).

5.1.3 The Probability of Events

So far, we have described the probability of each outcome with a non-negative number. This number represents the relative frequency of the outcome. Because we can tell when an event has occurred, we can compute the relative frequency of events, too. Because it is a relative frequency, the probability of an event is a non-negative number, and is no greater than one. But the probability of events must be consistent with the probability of outcomes. This implies a set of quite straightforward properties:

- **The probability of every event is non-negative**, which we write $P(\mathcal{A}) \geq 0$ for all $\mathcal{A}$ in the collection of events.
- **Every experiment has an outcome**, which we write $P(\Omega) = 1$.
- **The probability of disjoint events is additive**, which requires more notation. Assume that we have a collection of events $\mathcal{A}_i$, indexed by i. We require that these have the property $\mathcal{A}_i \cap \mathcal{A}_j = \emptyset$ when $i \neq j$. This means that there is no outcome that appears in more than one $\mathcal{A}_i$. In turn, if we interpret probability as relative frequency, we must have that $P(\cup_i \mathcal{A}_i) = \sum_i P(\mathcal{A}_i)$.

Any function P taking events to numbers that has these properties is a probability. These very simple properties imply a series of other very important properties.

Useful Facts: 5.1 *The probability of events*

- $P(\mathcal{A}^c) = 1 - P(\mathcal{A})$
- $P(\emptyset) = 0$
- $P(\mathcal{A} - \mathcal{B}) = P(\mathcal{A}) - P(\mathcal{A} \cap \mathcal{B})$
- $P(\mathcal{A} \cup \mathcal{B}) = P(\mathcal{A}) + P(\mathcal{B}) - P(\mathcal{A} \cap \mathcal{B})$
- $P(\cup_1^n \mathcal{A}_i) = \sum_i P(\mathcal{A}_i) - \sum_{i<j} P(\mathcal{A}_i \cap \mathcal{A}_j) + \sum_{i<j<k} P(\mathcal{A}_i \cap \mathcal{A}_j \cap \mathcal{A}_k) + \ldots (-1)^{(n+1)} P(\mathcal{A}_1 \cap \mathcal{A}_2 \cap \ldots \cap \mathcal{A}_n)$

I prove each of these below. Looking at the useful facts should suggest a helpful analogy. Think about the probability of an event as the "size" of that event. This "size" is relative to Ω, which has "size" 1. I find this a good way to remember equations.

For example, $P(\mathcal{A}) + P(\mathcal{A}^c) = 1$ has to be true, by this analogy, because $\mathcal{A}$ and $\mathcal{A}^c$ don't overlap, and together make up all of Ω. Similarly, $P(\mathcal{A} - \mathcal{B}) = P(\mathcal{A}) - P(\mathcal{A} \cap \mathcal{B})$ is easily captured — the "size" of the part of $\mathcal{A}$ that isn't $\mathcal{B}$ is obtained by taking the "size" of $\mathcal{A}$ and subtracting the "size" of the part that is also in $\mathcal{B}$. Similarly, $P(\mathcal{A} \cup \mathcal{B}) = P(\mathcal{A}) + P(\mathcal{B}) - P(\mathcal{A} \cap \mathcal{B})$ says — you can get the "size" of $\mathcal{A} \cup \mathcal{B}$ by adding the two "sizes", then subtracting the "size" of the intersection because otherwise you would count these terms twice. Some people find Venn diagrams a useful way to keep track of this argument, and Figure 5.1 is for them.

Proposition: $P(\mathcal{A}^c) = 1 - P(\mathcal{A})$

Proof: $\mathcal{A}^c$ and $\mathcal{A}$ are disjoint, so that $P(\mathcal{A}^c \cup \mathcal{A}) = P(\mathcal{A}^c) + P(\mathcal{A}) = P(\Omega) = 1$.

Proposition: $P(\emptyset) = 0$

Proof: $P(\emptyset) = P(\Omega^c) = P(\Omega - \Omega) = 1 - P(\Omega) = 1 - 1 = 0$.

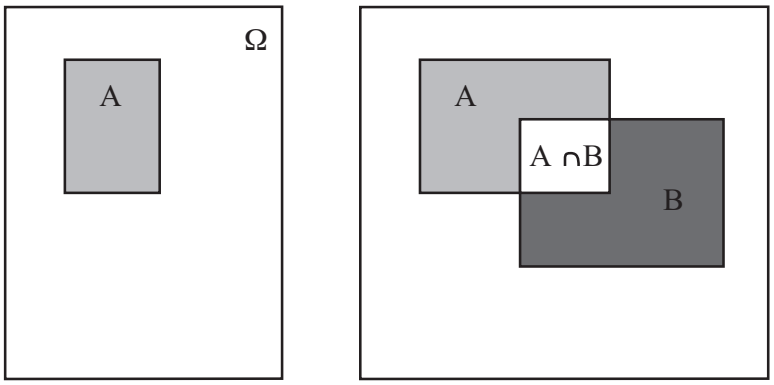

FIGURE 5.1: *If you think of the probability of an event as measuring its "size", many of the rules are quite straightforward to remember. Venn diagrams can sometimes help. On the* **left**, *a Venn diagram to help remember that* $P(\mathcal{A}) + P(\mathcal{A}^c) = 1$. *The "size" of* Ω *is 1, outcomes lie either in* $\mathcal{A}$ *or* $\mathcal{A}^c$, *and the two don't intersect. On the* **right**, *you can see that* $P(\mathcal{A} - \mathcal{B}) = P(\mathcal{A}) - P(\mathcal{A} \cap \mathcal{B})$ *by noticing that* $P(\mathcal{A} - \mathcal{B})$ *is the "size" of the part of* $\mathcal{A}$ *that isn't* $\mathcal{B}$. *This is obtained by taking the "size" of* $\mathcal{A}$ *and subtracting the "size" of the part that is also in* $\mathcal{B}$, *i.e. the "size" of* $\mathcal{A} \cap \mathcal{B}$. *Similarly, you can see that* $P(\mathcal{A} \cup \mathcal{B}) = P(\mathcal{A}) + P(\mathcal{B}) - P(\mathcal{A} \cap \mathcal{B})$ *by noticing that you can get the "size" of* $\mathcal{A} \cup \mathcal{B}$ *by adding the "sizes" of* $\mathcal{A}$ *and* $\mathcal{B}$, *then subtracting the "size" of the intersection to avoid double counting.*

Proposition: $P(\mathcal{A} - \mathcal{B}) = P(\mathcal{A}) - P(\mathcal{A} \cap \mathcal{B})$

Proof: $\mathcal{A} - \mathcal{B}$ is disjoint from $P(\mathcal{A} \cap \mathcal{B})$, and $(\mathcal{A} - \mathcal{B}) \cup (\mathcal{A} \cap \mathcal{B}) = \mathcal{A}$. This means that $P(\mathcal{A} - \mathcal{B}) + P(\mathcal{A} \cap \mathcal{B}) = P(\mathcal{A})$.

Proposition: $P(\mathcal{A} \cup \mathcal{B}) = P(\mathcal{A}) + P(\mathcal{B}) - P(\mathcal{A} \cap \mathcal{B})$

Proof: $P(\mathcal{A} \cup \mathcal{B}) = P(\mathcal{A} \cup (\mathcal{B} \cap \mathcal{A}^c)) = P(\mathcal{A}) + P((\mathcal{B} \cap \mathcal{A}^c))$. Now $\mathcal{B} = (\mathcal{B} \cap \mathcal{A}) \cup (\mathcal{B} \cap \mathcal{A}^c)$. Furthermore, $(\mathcal{B} \cap \mathcal{A})$ is disjoint from $(\mathcal{B} \cap \mathcal{A}^c)$, so we have $P(\mathcal{B}) = P((\mathcal{B} \cap \mathcal{A})) + P((\mathcal{B} \cap \mathcal{A}^c))$. This means that $P(\mathcal{A} \cup \mathcal{B}) = P(\mathcal{A}) + P((\mathcal{B} \cap \mathcal{A}^c)) = P(\mathcal{A}) + P(\mathcal{B}) - P((\mathcal{B} \cap \mathcal{A}))$.

Proposition: $P(\cup_1^n \mathcal{A}_i) = \sum_i P(\mathcal{A}_i) - \sum_{i<j} P(\mathcal{A}_i \cap \mathcal{A}_j) + \sum_{i<j<k} P(\mathcal{A}_i \cap \mathcal{A}_j \cap \mathcal{A}_k) + \ldots (-1)^{(n+1)} P(\mathcal{A}_1 \cap \mathcal{A}_2 \cap \ldots \cap \mathcal{A}_n)$

Proof: This can be proven by repeated application of the previous result. As an example, we show how to work the case where there are three sets (you can get the rest by induction).

$$\begin{aligned}
P(\mathcal{A}_1 \cup \mathcal{A}_2 \cup \mathcal{A}_3) &= P(\mathcal{A}_1 \cup (\mathcal{A}_2 \cup \mathcal{A}_3)) \\
&= P(\mathcal{A}_1) + P(\mathcal{A}_2 \cup \mathcal{A}_3) - P(\mathcal{A}_1 \cap (\mathcal{A}_2 \cup \mathcal{A}_3)) \\
&= P(\mathcal{A}_1) + (P(\mathcal{A}_2) + P(\mathcal{A}_3) - P(\mathcal{A}_2 \cap \mathcal{A}_3)) - \\
&\quad P((\mathcal{A}_1 \cap \mathcal{A}_2) \cup (\mathcal{A}_1 \cap \mathcal{A}_3)) \\
&= P(\mathcal{A}_1) + (P(\mathcal{A}_2) + P(\mathcal{A}_3) - P(\mathcal{A}_2 \cap \mathcal{A}_3)) - \\
&\quad P(\mathcal{A}_1 \cap \mathcal{A}_2) - P(\mathcal{A}_1 \cap \mathcal{A}_3) \\
&\quad -(-P((\mathcal{A}_1 \cap \mathcal{A}_2) \cap (\mathcal{A}_1 \cap \mathcal{A}_3))) \\
&= P(\mathcal{A}_1) + P(\mathcal{A}_2) + P(\mathcal{A}_3) - \\
&\quad P(\mathcal{A}_2 \cap \mathcal{A}_3) - P(\mathcal{A}_1 \cap \mathcal{A}_2) - P(\mathcal{A}_1 \cap \mathcal{A}_3) + \\
&\quad P(\mathcal{A}_1 \cap \mathcal{A}_2 \cap \mathcal{A}_3)
\end{aligned}$$

5.1.4 Computing Probabilities by Counting Outcomes

The rule $P(\mathcal{A} \cup \mathcal{B}) = P(\mathcal{A}) + P(\mathcal{B}) - P(\mathcal{A} \cap \mathcal{B})$ yields a very useful procedure for computing the probability of some events. Imagine $\mathcal{A}$ and $\mathcal{B}$ are disjoint (so $\mathcal{A} \cap \mathcal{B} = \emptyset$). Then $P(\mathcal{A} \cup \mathcal{B})$ is just $P(\mathcal{A}) + P(\mathcal{B})$. So imagine some event $\mathcal{E}$ that consists of a set of outcomes. Each singleton set — containing just one outcome — is disjoint from each other one. I can make the set $\mathcal{E}$ by: starting with an empty set; unioning this with one of the outcomes in $\mathcal{E}$; then repeatedly unioning the result a new outcome until I get $\mathcal{E}$. I am always unioning disjoint sets, so the probability of this event is the sum of probabilities of the outcomes. So we have

$$P(\mathcal{E}) = \sum_{O \in \mathcal{E}} P(O)$$

where O ranges over the outcomes in $\mathcal{E}$.

This is particularly useful when you know each outcome in Ω has the same probability. In this case, you can show

$$P(\mathcal{E}) = \frac{\text{Number of outcomes in } \mathcal{E}}{\text{Total number of outcomes in } \Omega}$$

(exercises). Such problems become, basically, advanced counting exercises.

Worked example 5.13 *Odd numbers with fair dice*

We throw a fair (each number has the same probability) die twice, then add the two numbers. What is the probability of getting an odd number?

Solution: There are 36 outcomes. Each has the same probability (1/36). 18 of them give an odd number, and the other 18 give an even number, so the probability is $18/36 = 1/2$

Worked example 5.14 *Drawing a red ten*

I shuffle a standard pack of cards, and draw one card. What is the probability that it is a red ten?

Solution: There are 52 cards, and each is an outcome. Two of these outcomes are red tens; so 1/26.

Worked example 5.15 *Numbers divisible by five with fair dice*

We throw a fair (each number has the same probability) die twice, then add the two numbers. What is the probability of getting a number divisible by five?

Solution: There are 36 outcomes. Each has the same probability (1/36). For this event, the spots must add to either 5 or to 10. There are 4 ways to get 5. There are 3 ways to get 10, so the probability is 7/36.

Worked example 5.16 *Children - 1*

This example is a version of of example 1.12, p44, in Stirzaker, "Elementary Probability". A couple decides to have children. They decide simply to have three children. Assume that each gender is equally likely at each birth. Let B_i be the event that there are i boys, and C be the event there are more girls than boys. Compute $P(B_1)$ and $P(C)$.

Solution: There are eight outcomes. Each has the same probability. Three of them have a single boy, so $P(B_1) = 3/8$. $P(C) = P(C^c)$ (because C^c is the event there are more boys than than girls, AND the number of children is odd), so that $P(C) = 1/2$; you can also get this by counting outcomes.

Sometimes a bit of fiddling with the space of outcomes makes it easy to

compute what we want.

Worked example 5.17 *Children - 2*

This example is a version of of example 1.12, p44, in Stirzaker, "Elementary Probability". A couple decides to have children. They decide to have children until the first girl is born, or until there are three, and then stop. Assume that each gender is equally likely at each birth. Let B_i be the event that there are i boys, and C be the event there are more girls than boys. Compute $P(B_1)$ and $P(C)$.

Solution: In this case, we could write the outcomes as $\{G, BG, BBG\}$, but if we think about them like this, we have no simple way to compute their probability. Instead, we could use the sample space from the previous answer, but assume that some of the later births are fictitious. This gives us natural collection of events for which it is easy to compute probabilities. Having one girl corresponds to the event $\{Gbb, Gbg, Ggb, Ggg\}$, where I have used lowercase letters to write the fictitious later births; the probability is $1/2$. Having a boy then a girl corresponds to the event $\{BGb, BGg\}$ (and so has probability $1/4$). Having two boys then a girl corresponds to the event $\{BBG\}$ (and so has probability $1/8$). Finally, having three boys corresponds to the event $\{BBB\}$ (and so has probability $1/8$). This means that $P(B_1) = 1/4$ and $P(C) = 1/2$.

Worked example 5.18 *Children - 2*

This example is a version of of example 1.12, p44, in Stirzaker, "Elementary Probability". A couple decides to have children. They decide to have children until there is one of each gender, or until there are three, and then stop. Assume that each gender is equally likely at each birth. Let B_i be the event that there are i boys, and C be the event there are more girls than boys. Compute $P(B_1)$ and $P(C)$.

Solution: We could write the outcomes as $\{GB, BG, GGB, GGG, BBG, BBB\}$. Again, if we think about them like this, we have no simple way to compute their probability; so we use the sample space from the previous example with the device of the fictitious births again. The important events are $\{GBb, GBg\}$; $\{BGb, BGg\}$; $\{GGB\}$; $\{GGG\}$; $\{BBG\}$; and $\{BBB\}$. Like this, we get $P(B_1) = 5/8$ and $P(C) = 1/4$.

Worked example 5.19 *Birthdays in succession*

We stop three people at random, and ask the day of the week on which they are born. What is the probability that they are born on three days of the week in succession (for example, the first on Monday; the second on Tuesday; the third on Wednesday; or Saturday-Sunday-Monday; and so on).

Solution: We assume that births are equally common on each day of the week. The space of outcomes consists of triples of days, and each outcome has the same probability. The event is the set of triples of three days in succession (which has seven elements, one for each starting day). The space of outcomes has 7^3 elements in it, so the probability is

$$\frac{\text{Number of outcomes in the event}}{\text{Total number of outcomes}} = \frac{7}{7^3} = \frac{1}{49}.$$

Worked example 5.20 *Shared birthdays*

We stop two people at random. What is the probability that they were born on the same day of the week?

Solution: The day the first person was born doesn't matter; the probability the second person was born on that day is 1/7. Or you could count outcomes explicitly to get

$$\frac{\text{Number of outcomes in the event}}{\text{Total number of outcomes}} = \frac{7}{7 \times 7} = \frac{1}{7}.$$

An important feature of this class of problem is that your intuition can be quite misleading. This is because, although each outcome can have very small probability, the number of outcomes in an event can be big.

5.1.5 Computing Probabilities by Reasoning about Sets

The rule $P(\mathcal{A}^c) = 1 - P(\mathcal{A})$ is occasionally useful for computing probabilities on its own; more commonly, you need other reasoning as well.

Worked example 5.21 *Shared birthdays*

What is the probability that, in a room of 30 people, there is a pair of people who have the same birthday?

Solution: We simplify, and assume that each year has 365 days, and that none of them are special (i.e. each day has the same probability of being chosen as a birthday). This model isn't perfect (there tend to be slightly more births roughly 9 months after: the start of spring; blackouts; major disasters; and so on) but it's workable. The easy way to attack this question is to notice that our probability, $P(\{\text{shared birthday}\})$, is

$$1 - P(\{\text{all birthdays different}\}).$$

This second probability is rather easy to compute. Each outcome in the sample space is a list of 30 days (one birthday per person). Each outcome has the same probability. So

$$P(\{\text{all birthdays different}\}) = \frac{\text{Number of outcomes in the event}}{\text{Total number of outcomes}}.$$

The total number of outcomes is easily seen to be 365^{30}, which is the total number of possible lists of 30 days. The number of outcomes in the event is the number of lists of 30 days, all different. To count these, we notice that there are 365 choices for the first day; 364 for the second; and so on. So we have

$$P(\{\text{shared birthday}\}) = 1 - \frac{365 \times 364 \times \ldots 336}{365^{30}} = 1 - 0.2937 = 0.7063$$

which means there's really a pretty good chance that two people in a room of 30 share a birthday.

If we change the birthday example slightly, the problem changes drastically. If you stand up in a room of 30 people and bet that two people in the room have the same birthday, you have a probability of winning of about 0.71. If you bet that there is someone else in the room who has the same birthday that you do, your probability of winning is very different.

Worked example 5.22 *Shared birthdays*

You bet there is someone else in a room of 30 people who has the same birthday that you do. Assuming you know nothing about the other 29 people, what is the probability of winning?

Solution: The easy way to do this is

$$P(\{\text{winning}\}) = 1 - P(\{\text{losing}\}).$$

Now you will lose if everyone has a birthday different from you. You can think of the birthdays of the others in the room as a list of 29 days of the year. If your birthday is on the list, you win; if it's not, you lose. The number of losing lists is the number of lists of 29 days of the year such that your birthday is not in the list. This number is easy to get. We have 364 days of the year to choose from for each of 29 locations in the list. The total number of lists is the number of lists of 29 days of the year. Each list has the same probability. So

$$P(\{\text{losing}\}) = \frac{364^{29}}{365^{29}}$$

and

$$P(\{\text{winning}\}) \approx 0.0765.$$

There is a wide variety of problems like this; if you're so inclined, you can make a small but quite reliable profit off people's inability to estimate probabilities for this kind of problem correctly (examples 21 and 22 are reliably profitable; you could probably do quite well out of examples 19 and 20).

The rule $P(\mathcal{A}-\mathcal{B}) = P(\mathcal{A})-P(\mathcal{A}\cap\mathcal{B})$ is also occasionally useful for computing probabilities on its own; more commonly, you need other reasoning as well.

Worked example 5.23 *Dice*

You flip two fair six-sided dice, and add the number of spots. What is the probability of getting a number divisible by 2, but not by 5?

Solution: There is an interesting way to work the problem. Write $\mathcal{D}_n$ for the event the number is divisible by n. Now $P(\mathcal{D}_2) = 1/2$ (count the cases; or, more elegantly, notice that each die has the same number of odd and even faces, and work from there). Now $P(\mathcal{D}_2 - \mathcal{D}_5) = P(\mathcal{D}_2) - P(\mathcal{D}_2 \cap \mathcal{D}_5)$. But $\mathcal{D}_2 \cap \mathcal{D}_5$ contains only two outcomes $(6, 4$ and $4, 6)$, so $P(\mathcal{D}_2 - \mathcal{D}_5) = 18/36 - 2/36 = 4/9$

Sometimes it is easier to reason about unions than to count outcomes directly.

Worked example 5.24 *Two fair dice*

I roll two fair dice. What is the probability that the result is divisible by either 2 or 5, or both?

Solution: Write $\mathcal{D}_n$ for the event the number is divisible by n. We want $P(\mathcal{D}_2 \cup \mathcal{D}_5) = P(\mathcal{D}_2) + P(\mathcal{D}_5) - P(\mathcal{D}_2 \cap \mathcal{D}_5)$. From example 23, we know $P(\mathcal{D}_2) = 1/2$ and $P(\mathcal{D}_2 \cap \mathcal{D}_5) = 2/36$. By counting outcomes, $P(\mathcal{D}_5) = 6/36$. So $P(\mathcal{D}_2 \cup \mathcal{D}_5) = (18 + 6 - 2)/36 = 22/36$.

5.1.6 Independence

Some experimental results do not affect others. For example, if I flip a coin twice, whether I get heads on the first flip has no effect on whether I get heads on the second flip. As another example, I flip a coin; the outcome does not affect whether I get hit on the head by a falling apple later in the day. We refer to events with this property as independent.

Useful Facts: 5.2 *Independent events*

Two events $\mathcal{A}$ and $\mathcal{B}$ are **independent** if and only if

$$P(\mathcal{A} \cap \mathcal{B}) = P(A)P(B)$$

The "size" analogy helps motivate this expression. We think of $P(A)$ as the "size" of $\mathcal{A}$ relative to Ω, and so on. Now $P(\mathcal{A} \cap \mathcal{B})$ measures the "size" of $\mathcal{A} \cap \mathcal{B}$ — that is, the part of $\mathcal{A}$ that lies inside $\mathcal{B}$. But if $\mathcal{A}$ and $\mathcal{B}$ are independent, then the size of $\mathcal{A} \cap \mathcal{B}$ relative to $\mathcal{B}$ should be the same as the size of $\mathcal{A}$ relative to Ω (Figure 5.2). Otherwise, $\mathcal{B}$ affects $\mathcal{A}$, because $\mathcal{A}$ is more (or less) likely when $\mathcal{B}$ has occurred.

So for $\mathcal{A}$ and $\mathcal{B}$ to be independent, we must have

$$\text{"Size" of } \mathcal{A} = \frac{\text{"Size" of piece of } \mathcal{A} \text{ in } \mathcal{B}}{\text{"Size" of } \mathcal{B}},$$

or, equivalently,

$$P(\mathcal{A}) = \frac{P(\mathcal{A} \cap \mathcal{B})}{P(\mathcal{B})}$$

which yields our expression. Independence is important, because it is straightforward to compute probabilities for sequences of independent outcomes.

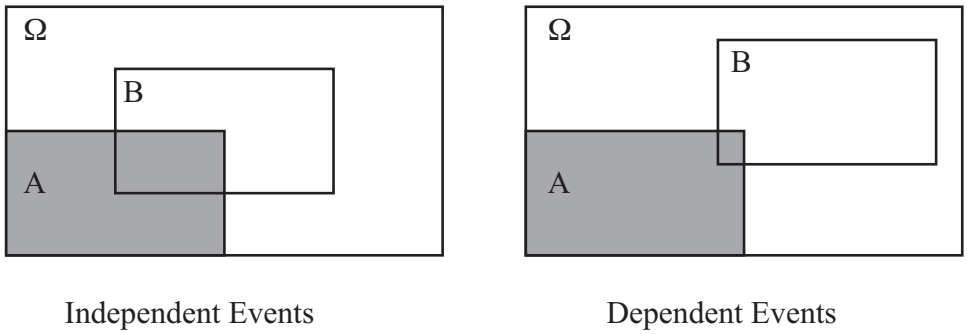

FIGURE 5.2: *On the* **left**, $\mathcal{A}$ *and* $\mathcal{B}$ *are independent.* $\mathcal{A}$ *spans* $1/4$ *of* Ω, *and* $\mathcal{A} \cap \mathcal{B}$ *spans* $1/4$ *of* $\mathcal{B}$. *This means that* $\mathcal{B}$ *can't affect* $\mathcal{A}$. $1/4$ *of the outcomes of* Ω *lie in* $\mathcal{A}$, *and* $1/4$ *of the outcomes in* $\mathcal{B}$ *lie in* $\mathcal{A} \cap \mathcal{B}$. *On the* **right**, *they are not. Very few of the outcomes in* $\mathcal{B}$ *lie in* $\mathcal{B} \cap \mathcal{A}$, *so that observing* $\mathcal{B}$ *means that* $\mathcal{A}$ *becomes less likely, because very few of the outcomes in* $\mathcal{B}$ *also lie in* $\mathcal{A} \cap \mathcal{B}$.

Worked example 5.25 *A fair coin*

A **fair** coin (or die, or whatever) is one where each outcome has the same probability. A fair coin has two sides, so the probability of each outcome must be $1/2$. We flip this coin twice - what is the probability we see two heads?

Solution: The flips are independent. Write $\mathcal{H}_1$ for the event that the first flip comes up heads, $\mathcal{H}_2$ for the event the second comes up heads. We have $\mathcal{H}_1 = \{HH, HT\}$ and $\mathcal{H}_2 = \{HH, TH\}$. We seek $P(\mathcal{H}_1 \cap \mathcal{H}_2) = P(\mathcal{H}_1)P(\mathcal{H}_2)$. The coin is fair, so $P(\mathcal{H}_1) = P(\mathcal{H}_2) = 1/2$. So the probability is $\frac{1}{4}$.

The reasoning of example 25 is moderately rigorous (if you want real rigor, you should specify the event space, etc.; I assumed that it would be obvious), but it's a bit clumsy for everyday use. The rule to remember is this: the probability that both events occur is $P(\mathcal{A} \cap \mathcal{B})$ and, if $\mathcal{A}$ and $\mathcal{B}$ are independent, then this is $P(\mathcal{A})P(\mathcal{B})$.

Worked example 5.26 *A fair die*

The space of outcomes for a fair six-sided die is

$$\{1, 2, 3, 4, 5, 6\}.$$

The die is fair means that each outcome has the same probability. Now we toss two fair dice — with what probability do we get two threes?

Solution:

$$\begin{aligned} P(\text{first toss yields 3} \cap \text{second toss yields 3}) &= P(\text{first toss yields 3}) \times \\ &\quad P(\text{second toss yields 3}) \\ &= (1/6)(1/6) \\ &= 1/36 \end{aligned}$$

Worked example 5.27 *Find the Lady, twice*

Assume that the card that is chosen is chosen fairly — that is, each card is chosen with the same probability. The game is played twice, and the cards are reshuffled between games. What is the probability of turning up a Queen and then a Queen again?

Solution: The events are independent, so 1/9.

Worked example 5.28 *Children*

A couple decides to have two children. Genders are assigned to children at random, fairly, independently and at birth (our models have to abstract a little!). What is the probability of having a boy and then a girl?

Solution:

$$\begin{aligned} P(\text{first is boy} \cap \text{second is girl}) &= P(\text{first is boy})P(\text{second is girl}) \\ &= (1/2)(1/2) \\ &= 1/4 \end{aligned}$$

You can use the expression to tell whether events are independent or not. Quite small changes to a problem affect whether events are independent. For ex-

ample, simply removing a card from a deck can make some events dependent.

Worked example 5.29 *Independent cards*

We shuffle a standard deck of 52 cards and draw one card. The event $\mathcal{A}$ is "the card is a red suit" and the event $\mathcal{B}$ is "the card is a 10". Are they independent?

Solution: $P(\mathcal{A}) = 1/2$, $P(\mathcal{B}) = 1/13$ and in example 14 we determined $P(\mathcal{A} \cap \mathcal{B}) = 2/52$. But $2/52 = 1/26 = P(\mathcal{A})P(\mathcal{B})$, so they are independent.

Worked example 5.30 *Independent cards*

We take a standard deck of cards, and remove the ten of hearts. We now shuffle this deck, and draw one card. The event $\mathcal{A}$ is "the card is a red suit" and the event $\mathcal{B}$ is "the card is a 10". Are they independent?

Solution: These are not independent because $P(\mathcal{A}) = 25/51$, $P(\mathcal{B}) = 3/51$ and $P(\mathcal{A} \cap \mathcal{B}) = 1/51 \neq P(\mathcal{A})P(\mathcal{B}) = 75/(51^2)$

The probability of a sequence of independent events can become very small very quickly, and this often misleads people.

Worked example 5.31 *Accidental DNA Matches*

I search a DNA database with a sample. Each time I attempt to match this sample to an entry in the database, there is a probability of an accidental chance match of $1e-4$. Chance matches are independent. There are 20, 000 people in the database. What is the probability I get at least one match, purely by chance?

Solution: This is $1 - P(\text{no chance matches})$. But $P(\text{no chance matches})$ is much smaller than you think. We have

$$\begin{aligned} P(\text{no chance matches}) &= P\left(\begin{array}{c} \text{no chance match to record } 1 \cap \\ \text{no chance match to record } 2 \cap \\ \ldots \cap \\ \text{no chance match to record } 20,000 \end{array}\right) \\ &= P(\text{no chance match to a record})^{20,000} \\ &= (1 - 1e-4)^{20,000} \end{aligned}$$

so the probability is about 0.86 that you get at least one match by chance. Notice that if the database gets bigger, the probability grows; so at 40, 000 the probability of one match by chance is 0.98.

5.1.7 Permutations and Combinations

Counting outcomes in an event can require pretty elaborate combinatorial arguments. One form of argument that is particularly important is to reason about permutations. You should recall that the number of permutations of k items is $k!$.

Worked example 5.32 *Counting outcomes with permutations*

I flip a coin k times. How many of the possible outcomes have exactly r heads?

Solution: Here is one natural way to think about this problem. Each outcome we are interested in is a string of r H's and $k-r$ T's, and we need to count the number of such strings. We can do so with permutations. Write down any string of r H's and $k-r$ T's. Any other string of r H's and $k-r$ T's is a permutation of this one. However, many of these permutations simply swap one H with another, or one T with another. The total number of permutations of a string of k entries is $k!$. The number of permutations that swap H's with one another is $r!$ (because there are r H's), and the number of permutations that swap T's with one another is $(k-r)!$. The total number of strings must then be

$$\frac{k!}{r!(k-r)!} = \left(\begin{array}{c} k \\ r \end{array} \right)$$

There is another way to think about example 32, which is more natural if you've seen combinations before. You start with a string of k T's, then you choose r distinct elements to turn into H's. The number of choices of r distinct elements in k items is:

$$\frac{k!}{r!(k-r)!} = \left(\begin{array}{c} k \\ r \end{array} \right),$$

so there must be this number of strings. We can use this result to investigate our model of probability as frequency.

Worked example 5.33 *A fair coin, revisited*

We conduct N experiments, where each experiment is to flip a fair coin twice. In what fraction of these experiments do we see both sides of the coin?

Solution: The sample space is $\{HH, HT, TH, TT\}$. The coin is fair, so each outcome has the same probability. This means that $(1/4)N$ experiments produce HT; $(1/4)N$ produce TH; and so on. We see both sides of the coin in an experiment for about $(1/2)N$ experiments.

Example 33 might seem like a contradiction to you. I claimed that we could

interpret P as the relative frequency of outcomes, and that my coin was fair; but, in only half of my experiments did I see both sides of the coin. In the other half, the coin behaved as if the probability of seeing one side is 1, and the probability of seeing the other side is 0. This occurs because the number of flips *in each experiment* is very small. If the number of flips is larger, you are much more likely to see about the right frequencies.

Worked example 5.34 *A fair coin, yet again*

We conduct N experiments, where each experiment is to flip a fair coin six times. In what fraction of these experiments do we get three heads and three tails?

Solution: There are $64 = 2^6$ outcomes in total for six coin flips. Each has the same probability. By the argument of example 32, there are

$$\frac{6!}{3!3!} = 20$$

outcomes with three heads. So the fraction is $20/64 = 1/3$.

At first glance, example 34 suggests that making the number of experiments get bigger doesn't help. But in the definition of probability, I said that there would be "about" $N \times P$ experiments with the outcome of interest.

Worked example 5.35 *A fair coin, yet again*

We conduct N experiments, where each experiment is to flip a fair coin 10 times. In what fraction of these experiments do we get between 4 and 6 heads?

Solution: There are $1024 = 2^{10}$ outcomes for an experiment. We are interested in the four head outcomes, the five head outcomes and the six head outcomes. Using the argument of example 34 gives

$$\begin{aligned}
\text{total outcomes} &= \text{4H outcomes} + \text{5H outcomes} + \text{6H outcomes} \\
&= \frac{10!}{4!6!} + \frac{10!}{5!5!} + \frac{10!}{6!4!} \\
&= 210 + 252 + 210 \\
&= 692
\end{aligned}$$

so in $692/1024 \approx 0.68$ of the experiments, we will see between four and six heads

Worked example 5.36 *A fair coin, and a lot of flipping*

We conduct N experiments, where each experiment is to flip a fair coin 100 times. In what fraction of these experiments do we get between 45 and 65 heads?

Solution: There are 2^{100} outcomes for an experiment. Using the argument of example 35 gives

$$\text{total outcomes} = \sum_{i=45}^{65} \frac{100!}{i!(100-i)!}$$

which isn't particularly easy to evaluate.

As these examples suggest, if an experiment consists of flipping a large number of coins, then a high fraction of those experiments will show heads with a frequency very close to the right number. We will establish this later, with rather more powerful machinery.

Worked example 5.37 *Overbooking - 1*

An airline has a regular flight with six seats. It always sells seven tickets. Passengers turn up for the flight with probability p, and do so independent of other passengers. What is the probability that the flight is overbooked?

Solution: This is like a coin-flip problem; think of each passenger as a biased coin. With probability p, it comes up T (for *t*urn up) and with probability $(1-p)$, it turns up N (for *n*o-show). This coin is flipped seven times, and we are interested in the probability that there are seven T's. This is p^7, because the flips are independent.

Worked example 5.38 *Overbooking - 2*

An airline has a regular flight with six seats. It always sells eight tickets. Passengers turn up for the flight with probability p, and do so independent of other passengers. What is the probability that six passengers arrive? (i.e. the flight is not overbooked or underbooked).

Solution: Now we flip the coin eight times, and are interested in the probability of getting exactly six T's. We get, by the reasoning of worked example 32,

$$\frac{8!}{2!6!}p^6(1-p)^2 = 28p^6(1-p)^2$$

Worked example 5.39 *Overbooking - 3*

An airline has a regular flight with six seats. It always sells eight tickets. Passengers turn up for the flight with probability p, and do so independent of other passengers. What is the probability that the flight is overbooked?

Solution: Now we flip the coin eight times, and are interested in the probability of getting more than six T's. This is the union of two disjoint events (seven T's and eight T's). For the case of seven T's, one flip must be N; there are eight choices. For the case of eight T's, all eight flips must be T, and there is only one way to achieve this. So the probability the flight is overbooked is

$$\begin{aligned} P(\text{overbooked}) &= P(7\ T\text{'s} \cup 8\ T\text{'s}) \\ &= P(7\ T\text{'s}) + P(8\ T\text{'s}) \\ &= 8p^7(1-p) + p^8 \end{aligned}$$

Worked example 5.40 *Overbooking - 4*

An airline has a regular flight with s seats. It always sells t tickets. Passengers turn up for the flight with probability p, and do so independent of other passengers. What is the probability that u passengers turn up?

Solution: Now we flip the coin t times, and are interested in the probability of getting u T's. By the argument of worked example 32, there are

$$\frac{t!}{u!(t-u)!}$$

disjoint outcomes with u T's and $t-u$ N's. Each such outcome is independent, and has probability $p^u(1-p)^{t-u}$. So

$$P(u \text{ passengers turn up}) = \frac{t!}{u!(t-u)!}p^u(1-p)^{t-u}$$

Worked example 5.41 *Overbooking - 5*

An airline has a regular flight with s seats. It always sells t tickets. Passengers turn up for the flight with probability p, and do so independent of other passengers. What is the probability that the flight is oversold?

Solution: We need $P(\{s+1 \text{ turn up}\} \cup \{s+2 \text{ turn up}\} \cup \ldots \cup \{t \text{ turn up}\})$. But the events $\{i \text{ turn up}\}$ and $\{j \text{ turn up}\}$ are disjoint if $i \neq j$. So we can exploit example 40, and write

$$\begin{aligned} P(\text{overbooked}) &= P(\{s+1 \text{ turn up}\}) + P(\{s+2 \text{ turn up}\}) + \\ &\quad \ldots P(\{t \text{ turn up}\}) \\ &= \sum_{i=s+1}^{t} P(\{i \text{ turn up}\}) \\ &= \sum_{i=s+1}^{t} \frac{t!}{i!(t-i)!}p^i(1-p)^{t-i} \end{aligned}$$

5.2 CONDITIONAL PROBABILITY

If you throw a fair die twice and add the numbers, then the probability of getting a number less than six is $\frac{10}{36}$. Now imagine you know that the first die came up

three. In this case, the probability that the sum will be less than six is $\frac{1}{3}$, which is slightly larger. If the first die came up four, then the probability the sum will be less than six is $\frac{1}{6}$, which is rather less than $\frac{10}{36}$. If the first die came up one, then the probability that the sum is less than six becomes $\frac{2}{3}$, which is much larger.

Each of these probabilities is an example of a **conditional probability**. We assume we have a space of outcomes and a collection of events. The conditional probability of $\mathcal{B}$, conditioned on $\mathcal{A}$, is the probability that $\mathcal{B}$ occurs given that $\mathcal{A}$ has definitely occurred. We write this as

$$P(\mathcal{B}|\mathcal{A})$$

5.2.1 Evaluating Conditional Probabilities

Now to get an expression for $P(\mathcal{B}|\mathcal{A})$, notice that, because $\mathcal{A}$ is known to have occurred, our space of outcomes or sample space is now reduced to $\mathcal{A}$. We know that our outcome lies in $\mathcal{A}$; $P(\mathcal{B}|\mathcal{A})$ is the probability that it also lies in $\mathcal{B} \cap \mathcal{A}$.

The outcome lies in $\mathcal{A}$, and so it must lie in either $P(\mathcal{B} \cap \mathcal{A})$ or in $P(\mathcal{B}^c \cap \mathcal{A})$. This means that

$$P(\mathcal{B}|\mathcal{A}) + P(\mathcal{B}^c|\mathcal{A}) = 1.$$

Now recall the idea of probabilities as relative frequencies. If $P(\mathcal{C} \cap \mathcal{A}) = kP(\mathcal{B} \cap \mathcal{A})$, this means that we will see outcomes in $\mathcal{C} \cap \mathcal{A}$ k times as often as we will see outcomes in $\mathcal{B} \cap \mathcal{A}$. But this must apply even if we know in advance that the outcome is in $\mathcal{A}$. So we must have

$$P(\mathcal{B}|\mathcal{A}) \propto P(\mathcal{B} \cap \mathcal{A}).$$

Now we need to determine the constant of proportionality; write c for this constant, meaning

$$P(\mathcal{B}|\mathcal{A}) = cP(\mathcal{B} \cap \mathcal{A}).$$

We have that

$$P(\mathcal{B}|\mathcal{A}) + P(\mathcal{B}^c|\mathcal{A}) = cP(\mathcal{B} \cap \mathcal{A}) + cP(\mathcal{B}^c \cap \mathcal{A}) = cP(\mathcal{A}) = 1,$$

so that

$$\boxed{P(\mathcal{B}|\mathcal{A}) = \frac{P(\mathcal{B} \cap \mathcal{A})}{P(\mathcal{A})}.}$$

Using the "size" metaphor, this says the probability that an outcome, *which is known to be in* $\mathcal{A}$, is also in $\mathcal{B}$ is the fraction of $\mathcal{A}$ that is also in $\mathcal{B}$.

Another, very useful, way to write this expression is

$$P(\mathcal{B}|\mathcal{A})P(\mathcal{A}) = P(\mathcal{B} \cap \mathcal{A}).$$

Now, since $\mathcal{B} \cap \mathcal{A} = \mathcal{A} \cap \mathcal{B}$, we must have that

$$P(\mathcal{B}|\mathcal{A}) = \frac{P(\mathcal{A}|\mathcal{B})P(\mathcal{B})}{P(\mathcal{A})}$$

Worked example 5.42 *Two dice*

We throw two fair dice. What is the conditional probability that the sum of spots on both dice is greater than six, conditioned on the event that the first die comes up five?

Solution: Write the event that the first die comes up 5 as $\mathcal{F}$, and the event the sum is greather than six as $\mathcal{S}$. There are five outcomes where the first die comes up 5 and the number is greater than 6, so $P(\mathcal{F} \cap \mathcal{S}) = 5/36$. $P(\mathcal{S}|\mathcal{F}) = P(\mathcal{F} \cap \mathcal{S})/P(\mathcal{F}) = (5/36)/(1/6) = 5/6$.

Notice that $\mathcal{A} \cap \mathcal{B}$ and $\mathcal{A} \cap \mathcal{B}^c$ are disjoint sets, and that $\mathcal{A} = (\mathcal{A} \cap \mathcal{B}) \cup (\mathcal{A} \cap \mathcal{B}^c)$. So, because $P(\mathcal{A}) = P(\mathcal{A} \cap \mathcal{B}) + P(\mathcal{A} \cap \mathcal{B}^c)$, we have

$$P(\mathcal{A}) = P(\mathcal{A}|\mathcal{B})P(\mathcal{B}) + P(\mathcal{A}|\mathcal{B}^c)P(\mathcal{B}^c)$$

a tremendously important and useful fact. Another version of this fact is also very useful. Assume we have a set of disjoint sets $\mathcal{B}_i$. These sets must have the property that (a) $\mathcal{B}_i \cap \mathcal{B}_j = \emptyset$ for $i \neq j$ and (b) they cover $\mathcal{A}$, meaning that $\mathcal{A} \cap (\cup_i \mathcal{B}_i) = \mathcal{A}$. Then, because $P(\mathcal{A}) = \sum_i P(\mathcal{A} \cap \mathcal{B}_i)$, so we have

$$P(\mathcal{A}) = \sum_i P(\mathcal{A}|\mathcal{B}_i)P(\mathcal{B}_i)$$

Worked example 5.43 *Car factories*

There are two car factories, A and B. Each year, factory A produces 1000 cars, of which 10 are lemons. Factory B produces 2 cars, each of which is a lemon. All cars go to a single lot, where they are thoroughly mixed up. I buy a car.

- What is the probability it is a lemon?
- What is the probability it came from factory B?
- The car is now revealed to be a lemon. What is the probability it came from factory B, conditioned on the fact it is a lemon?

Solution:

- Write the event the car is a lemon as $\mathcal{L}$. There are 1002 cars, of which 12 are lemons. The probability that I select any given car is the same, so we have 12/1002.
- Same argument yields 2/1002.
- Write $\mathcal{B}$ for the event the car comes from factory B. I need $P(\mathcal{B}|\mathcal{L})$. This is $P(\mathcal{L}|\mathcal{B})P(\mathcal{B})/P(\mathcal{L}) = (1 \times 2/1002)/(12/1002) = 1/6$.

Worked example 5.44 *Royal flushes in poker - 1*

This exercise is after Stirzaker, p. 51.
You are playing a straightforward version of poker, where you are dealt five cards face down. A royal flush is a hand of AKQJ10 all in one suit. What is the probability that you are dealt a royal flush?

Solution: This is

$$\frac{\text{number of hands that are royal flushes, ignoring card order}}{\text{total number of different five card hands, ignoring card order}}.$$

There are four hands that are royal flushes (one for each suit). Now the total number of five card hands is

$$\left(\begin{array}{c} 52 \\ 5 \end{array} \right) = 2598960$$

so we have

$$\frac{4}{2598960} = \frac{1}{649740}.$$

Worked example 5.45 *Royal flushes in poker - 2*

This exercise is after Stirzaker, p. 51.
You are playing a straightforward version of poker, where you are dealt five cards face down. A royal flush is a hand of AKQJ10 all in one suit. The fifth card that you are dealt lands face up. It is the nine of spades. What now is the probability that your have been dealt a royal flush? (i.e. what is the conditional probability of getting a royal flush, conditioned on the event that one card is the nine of spades)

Solution: No hand containing a nine of spades is a royal flush, so this is easily zero.

Worked example 5.46 *Royal flushes in poker - 3*

This exercise is after Stirzaker, p. 51.
You are playing a straightforward version of poker, where you are dealt five cards face down. A royal flush is a hand of AKQJ10 all in one suit. The fifth card that you are dealt lands face up. It is the Ace of spades. What now is the probability that your have been dealt a royal flush? (i.e. what is the conditional probability of getting a royal flush, conditioned on the event that one card is the Ace of spades)

Solution: There are two ways to do this. The easiest is to notice that the answer is the probability that the other four cards are KQJ10 of spades, which is

$$1/\left(\begin{array}{c}51\\4\end{array}\right) = \frac{1}{249900}.$$

Harder is to consider the events

$\mathcal{A}$ = event that you receive a royal flush and last card is the ace of spades

and

$\mathcal{B}$ = event that the last card you receive is the ace of spades,

and the expression

$$P(\mathcal{A}|\mathcal{B}) = \frac{P(\mathcal{A}\cap\mathcal{B})}{P(\mathcal{B})}.$$

Now $P(\mathcal{A}) = \frac{1}{52}$. $P(\mathcal{A}\cap\mathcal{B})$ is given by

$$\frac{\text{number of five card royal flushes where card five is Ace of spades}}{\text{total number of different five card hands}}.$$

where we DO NOT ignore card order. This is

$$\frac{4\times 3\times 2\times 1}{52\times 51\times 50\times 49\times 48}$$

yielding

$$P(\mathcal{A}|\mathcal{B}) = \frac{1}{249900}.$$

Notice the interesting part: the conditional probability is rather larger than the probability. If you see this ace, the conditional probability is $\frac{13}{5}$ times the probability that you will get a flush if you don't. Seeing this card has really made a difference.

Worked example 5.47 *False positives*

After Stirzaker, p55. You have a blood test for a rare disease that occurs by chance in 1 person in 100, 000. If you have the disease, the test will report that you do with probability 0.95 (and that you do not with probability 0.05). If you do not have the disease, the test will report a false positive with probability 1e-3. If the test says you do have the disease, what is the probability it is correct?

Solution: Write S for the event you are sick and R for the event the test reports you are sick. We need $P(S|R)$. We have

$$\begin{aligned} P(S|R) &= \frac{P(R|S)P(S)}{P(R)} \\ &= \frac{P(R|S)P(S)}{P(R|S)P(S) + P(R|S^c)P(S^c)} \\ &= \frac{0.95 \times 1e-5}{0.95 \times 1e-5 + 1e-3 \times (1 - 1e-5)} \\ &= 0.0094 \end{aligned}$$

which should strike you as being a bit alarming. The disease is so rare that the test is almost useless.

Worked example 5.48 *False positives -2*

After Stirzaker, p55. You want to *design* a blood test for a rare disease that occurs by chance in 1 person in 100, 000. If you have the disease, the test will report that you do with probability p (and that you do not with probability $(1-p)$). If you do not have the disease, the test will report a false positive with probability q. You want to choose the value of p so that if the test says you have the disease, there is at least a 50% probability that you do.

Solution: Write S for the event you are sick and R for the event the test reports you are sick. We need $P(S|R)$. We have

$$\begin{aligned} P(S|R) &= \frac{P(R|S)P(S)}{P(R)} \\ &= \frac{P(R|S)P(S)}{P(R|S)P(S) + P(R|S^c)P(S^c)} \\ &= \frac{p \times 1e-5}{p \times 1e-5 + q \times (1-1e-5)} \\ &\geq 0.5 \end{aligned}$$

which means that $p \geq 99999q$ which should strike you as being very alarming indeed, because $p \leq 1$ and $q \geq 0$. One plausible pair of values is $q = 1e-5$, $p = 1-1e-5$. The test has to be spectacularly accurate to be of any use.

5.2.2 The Prosecutors Fallacy

It is quite easy to make mistakes in conditional probability. Several such mistakes have names, because they're so common. One is the **prosecutor's fallacy**. This often occurs in the following form: A prosecutor has evidence $\mathcal{E}$ against a suspect. Write $\mathcal{I}$ for the event that the suspect is innocent. The evidence has the property that $P(\mathcal{E}|\mathcal{I})$ is extremely small. The prosecutor argues that the suspect must be guilty, because $P(\mathcal{E}|\mathcal{I})$ is so small, and this is the fallacy.

The problem here is that the conditional probability of interest is $P(\mathcal{I}|\mathcal{E})$ (rather than $P(\mathcal{E}|\mathcal{I})$). The fact that $P(\mathcal{E}|\mathcal{I})$ is small doesn't mean that $P(\mathcal{I}|\mathcal{E})$ is small, because

$$P(\mathcal{I}|\mathcal{E}) = \frac{P(\mathcal{E}|\mathcal{I})P(\mathcal{I})}{P(\mathcal{E})} = \frac{P(\mathcal{E}|\mathcal{I})P(\mathcal{I})}{(P(\mathcal{E}|\mathcal{I})P(\mathcal{I}) + P(\mathcal{E}|\mathcal{I}^c)(1-P(\mathcal{I})))}$$

Notice how, if $P(\mathcal{I})$ is large or if $P(\mathcal{E}|\mathcal{I}^c)$ is much smaller than $P(\mathcal{E}|\mathcal{I})$, then $P(\mathcal{I}|\mathcal{E})$ could be close to one. The question to look at is not how unlikely the evidence is if the subject is innocent; instead, the question is how likely the subject is to be guilty compared to some other source of the evidence. These are two very different questions.

One useful analogy may be helpful. If you buy a lottery ticket ($\mathcal{L}$), the probability of winning ($\mathcal{W}$) is small. So $P(\mathcal{W}|\mathcal{L})$ may be very small. But $P(\mathcal{L}|\mathcal{W})$ is 1 — the winner is always someone who bought a ticket.

The prosecutor's fallacy has contributed to a variety of miscarriages of justice. One famous incident involved a mother, Sally Clark, convicted of murdering two of her children. Expert evidence by paediatrician Roy Meadow argued that the probability of both deaths resulting from Sudden Infant Death Syndrome was extremely small. Her first appeal cited, among other grounds, statistical error in the evidence (you should spot the prosecutors fallacy; others were involved, too). The appeals court rejected this appeal, calling the statistical point "a sideshow". This prompted a great deal of controversy, both in the public press and various professional journals, including a letter from the then president of the Royal Statistical Society to the Lord Chancellor, pointing out that "*statistical evidence ... (should be) ... presented only by appropriately qualified statistical experts*". A second appeal (on other grounds) followed, and was successful. The appelate judges specifically criticized the statistical evidence, although it was not a point of appeal. Clark never recovered from this horrific set of events and died in tragic circumstances shortly after the second appeal. Roy Meadow was then struck off the rolls for serious professional misconduct as an expert witness, a ruling he appealed successfully. You can find a more detailed account of this case, with pointers to important documents including the letter to the Lord Chancellor (which is well worth reading), at `http://en.wikipedia.org/wiki/Roy_Meadow`; there is further material on the prosecutors fallacy at `http://en.wikipedia.org/wiki/Prosecutor\%27s_fallacy`.

5.2.3 Independence and Conditional Probability

As we have seen, two events are **independent** if

$$P(\mathcal{A} \cap \mathcal{B}) = P(\mathcal{A})P(\mathcal{B}).$$

If two events $\mathcal{A}$ and $\mathcal{B}$ are independent, then

$$P(\mathcal{A}|\mathcal{B}) = P(\mathcal{A})$$

and

$$P(\mathcal{B}|\mathcal{A}) = P(\mathcal{B}).$$

Again, this means that knowing that $\mathcal{A}$ occurred tells you nothing about $\mathcal{B}$ — the probability that $\mathcal{B}$ will occur is the same whether you know that $\mathcal{A}$ occurred or not.

Events $\mathcal{A}_1 \ldots \mathcal{A}_n$ are **pairwise independent** if each pair is independent (i.e. $\mathcal{A}_1$ and $\mathcal{A}_2$ are independent, etc.). They are **independent** if for any collection of distinct indices $i_1 \ldots i_k$ we have

$$P(\mathcal{A}_{i_1} \cap \ldots \cap \mathcal{A}_{i_k}) = P(\mathcal{A}_{i_1}) \ldots P(\mathcal{A}_{i_k})$$

Notice that independence is a much stronger assumption than pairwise independence.

Worked example 5.49 *Cards and pairwise independence*

We draw three cards from a properly shuffled standard deck, with replacement and reshuffling (i.e., draw a card, make a note, return to deck, shuffle, draw the next, make a note, shuffle, draw the third). Let $\mathcal{A}$ be the event that "card 1 and card 2 have the same suit"; let $\mathcal{B}$ be the event that "card 2 and card 3 have the same suit"; let $\mathcal{C}$ be the event that "card 1 and card 3 have the same suit". Show these events are pairwise independent, but not independent.

Solution: By counting, you can check that $P(\mathcal{A}) = 1/4$; $P(\mathcal{B}) = 1/4$; and $P(\mathcal{A} \cap \mathcal{B}) = 1/16$, so that these two are independent. This argument works for other pairs, too. But $P(\mathcal{C} \cap \mathcal{A} \cap \mathcal{B}) = 1/16$ which is not $1/4^3$, so the events are not independent; this is because the third event is logically implied by the first two.

We usually do not have the information required to prove that events are independent. Instead, we use intuition (for example, two flips of the same coin are likely to be independent unless there is something very funny going on) or simply choose to apply models in which some variables are independent.

Some events are pretty obviously independent. On other occasions, one needs to think about whether they are independent or not. Sometimes, it is reasonable to choose to model events as being independent, even though they might not be exactly independent. In cases like this, it is good practice to state your assumptions, because it helps you to keep track of what your model means. For example, we have worked with the event that a person, selected fairly and randomly from a set of people in a room, has a birthday on a particular day of the year. We assumed that, for different people, the events are independent. This seems like a fair assumption, but one might want to be cautious if you know that the people in the room are drawn from a population where multiple births are common.

Independent events can lead very quickly to very small probabilities, as we saw in example 31. This can mislead intuition quite badly, and lead to serious problems. In particular, these small probabilities can interact with the prosecutor's fallacy in a dangerous way. In example 31, we saw how the probability of getting a chance match in a large DNA database could be quite big, even though the probability of a single match is small. One version of the prosecutors fallacy is to argue that, because the probability of a single match is small, the person who matched the DNA must have committed the crime. The fallacy is to ignore the fact that the probability of a chance match to a large database is quite high.

People quite often reason poorly about independent events. The most common problem is known as the **gambler's fallacy**. This occurs when you reason that the probability of an independent event has been changed by previous outcomes. For example, imagine I toss a coin that is known to be fair 20 times and get 20 heads. The probability that the next toss will result in a head has not changed at all — it is still 0.5 — but many people will believe that it has changed. This idea is also sometimes referred to as **antichance**.

It might in fact be sensible to behave as if you're committing some version of the gambler's fallacy in real life, because you hardly ever know for sure that your model is right. So in the coin tossing example, if the coin wasn't known to be fair, it might be reasonable to assume that it has been weighted in some way, and so to believe that the more heads you see, the more likely you will see a head in the next toss. At time of writing, Wikipedia has some fascinating stories about the gambler's fallacy; apparently, in 1913, a roulette wheel in Monte Carlo produced black 26 times in a row, and gamblers lost an immense amount of money betting on red. Here the gambler's reasoning seems to have been that the universe should ensure that probabilities produce the right frequencies in the end, and so will adjust the outcome of the next spin of the wheel to balance the sums. This is an instance of the gambler's fallacy. However, the page also contains the story of one Joseph Jagger, who hired people to keep records of the roulette wheels, and notice that one wheel favored some numbers (presumably because of some problem with balance). He won a lot of money, until the casino started more careful maintenance on the wheels. This isn't the gambler's fallacy; instead, he noticed that the numbers implied that the wheel was not a fair randomizer. He made money because the casino's odds on the bet assumed that it was fair.

5.3 EXAMPLE: THE MONTY HALL PROBLEM

Careless thinking about probability, particularly conditional probability, can cause wonderful confusion. The Monty Hall problem is a relatively simple exercise in conditional probability. Nonetheless, it has been the subject of extensive, lively, and often quite inaccurate correspondence in various national periodicals — it seems to catch the attention, which is why we describe it in some detail. The problem works like this: There are three doors. Behind one is a car. Behind each of the others is a goat. The car and goats are placed randomly and fairly, so that the probability that there is a car behind each door is the same. You will get the object that lies behind the door you choose at the end of the game. The goats are interchangeable, and, for reasons of your own, you would prefer the car to a goat.

The game goes as follows. You select a door. The host then opens a door and shows you a goat. You must now choose to either keep your door, or switch to the other door. What should you do?

You cannot tell what to do, by the following argument. Label the door you chose at the start of the game 1; the other doors 2 and 3. Write C_i for the event that the car lies behind door i. Write G_m for the event that a goat is revealed behind door m, where m is the number of the door where the goat was revealed (which could be 1, 2, or 3). You need to know $P(C_1|G_m)$. But

$$P(C_1|G_m) = \frac{P(G_m|C_1)P(C_1)}{P(G_m|C_1)P(C_1) + P(G_m|C_2)P(C_2) + P(G_m|C_3)P(C_3)}$$

and you do not know $P(G_m|C_1)$, $P(G_m|C_2)$, $P(G_m|C_3)$, because you don't know the rule by which the host chooses which door to open to reveal a goat. Different rules lead to quite different analyses.

There are several possible rules for the host to show a goat:

- **Rule 1:** choose a door uniformly at random.

- **Rule 2:** choose from the doors with goats behind them *that are not door 1* uniformly and at random.
- **Rule 3:** if the car is at 1, then choose 2; if at 2, choose 3; if at 3, choose 1.
- **Rule 4:** choose from the doors with goats behind them uniformly and at random.

We should keep track of the rules in the conditioning, so we write $P(G_m|C_1, r_1)$ for the conditional probability that a goat was revealed behind door m when the car is behind door 1, using rule 1 (and so on). This means we are interested in

$$P(C_1|G_m, r_n) = \frac{P(G_m|C_1, r_n)P(C_1)}{P(G_m|C_1, r_n)P(C_1) + P(G_m|C_2, r_n)P(C_2) + P(G_m|C_3, r_n)P(C_3)}.$$

Worked example 5.50 *Monty Hall, rule one*

Assume the host uses rule one, and shows you a goat behind door two. What is $P(C_1|G_2, r_1)$?

Solution: To work this out, we need to know $P(G_2|C_1, r_1)$, $P(G_2|C_2, r_1)$ and $P(G_2|C_3, r_1)$. Now $P(G_2|C_2, r_1)$ must be zero, because the host could not reveal a goat behind door two if there was a car behind that door. Write O_2 for the event the host *chooses* to open door two, and B_2 for the event there happens to be a goat behind door two. These two events are independent — the host chose the door uniformly at random. We can compute

$$\begin{aligned} P(G_2|C_1, r_1) &= P(O_2 \cap B_2|C_1, r_1) \\ &= P(O_2|C_1, r_1)P(B_2|C_1, r_1) \\ &= (1/3)(1) \\ &= 1/3 \end{aligned}$$

where $P(B_2|C_1, r_1) = 1$ because we conditioned on the fact there was a car behind door one, so there is a goat behind each other door. This argument establishes $P(G_2|C_3, r_1) = 1/3$, too. So $P(C_1|G_2, r_1) = 1/2$ — the host showing you the goat does not motivate you to do anything, because if $P(C_1|G_2, r_1) = 1/2$, then $P(C_3|G_2, r_1) = 1/2$, too — there's nothing to choose between the two closed doors.

Worked example 5.51 *Monty Hall, rule two*

Assume the host uses rule two, and shows you a goat behind door two. What is $P(C_1|G_2, r_2)$?

Solution: To work this out, we need to know $P(G_2|C_1, r_2)$, $P(G_2|C_2, r_2)$ and $P(G_2|C_3, r_2)$. Now $P(G_2|C_2, r_2) = 0$, because the host chooses from doors with goats behind them. $P(G_2|C_1, r_2) = 1/2$, because the host chooses uniformly and at random from doors with goats behind them that are not door one; if the car is behind door one, there are two such doors. $P(G_2|C_3, r_2) = 1$, because there is only one door that (a) has a goat behind it and (b) isn't door one. Plug these numbers into the formula, to get $P(C_1|G_2, r_2) = 1/3$. This is the source of all the fuss. It says that, if you know the host is using rule two, you should switch doors if the host shows you a goat behind door two (because $P(C_3|G_2, r_2) = 2/3$).

Notice what is happening: if the car is behind door three, then the *only* choice of goat for the host is the goat behind two. So by choosing a door under rule two, the host is signalling some information to you, which you can use. By using rule three, the host can tell you precisely where the car is (exercises).

Many people find the result of example 51 counterintuitive, and object (sometimes loudly, in newspaper columns, letters to the editor, etc.). One example that some people find helpful is an extreme case. Imagine that, instead of three doors, there are 1002. The host is using rule two, modified in the following way: open all but one of the doors that are not door one, choosing only doors that have goats behind them to open. You choose door one; the host opens 1000 doors — say, all but doors one and 1002. What would you do?

5.4 WHAT YOU SHOULD REMEMBER

You should be able to:

- Write out a set of outcomes for an experiment.
- Construct an event space.
- Compute the probabilities of outcomes and events.
- Determine when events are independent.
- Compute the probabilities of outcomes by counting events, when the count is straightforward.
- Compute a conditional probability.

You should remember:

- The definition of an event space.
- The properties of a probability function.

- The definition of independence.
- The definition of conditional probability.

PROBLEMS

Outcomes

5.1. You roll a four sided die. What is the space of outcomes?

5.2. King Lear decides to allocate three provinces (1, 2, and 3) to his daughters (Goneril, Regan and Cordelia - read the book) at random. Each gets one province. What is the space of outcomes?

5.3. You randomly wave a flyswatter at a fly. What is the space of outcomes?

5.4. You read the book, so you know that King Lear had family problems. As a result, he decides to allocate two provinces to one daughter, one province to another daughter, and no provinces to the third. Because he's a bad problem solver, he does so at random. What is the space of outcomes?

Probability of outcomes

5.5. You roll a fair four sided die. What is the probability of getting a 3?

5.6. You roll a fair four sided die, and then a fair six sided die. You add the numbers on the two dice. What is the probability the result is even?

5.7. You roll a fair 20 sided die. What is the probability of getting an even number?

5.8. You roll a fair five sided die. What is the probability of getting an even number?

Events

5.9. At a particular University, 1/2 of the students drink alcohol and 1/3 of the students smoke cigarettes.

(a) What is the largest possible fraction of students who do neither?

(b) It turns out that, in fact, 1/3 of the students do neither. What fraction of the students does both?

5.10. I flip two coins. What one set needs to be added to this collection of sets to form an event space?

$$\Sigma = \{\emptyset, \Omega, \{TH\}, \{HT, TH, TT\}, \{HH\}, \{HT, TT\}, \{HH, TH\}\}$$

Probability of Events

5.11. Assume each outcome in Ω has the same probability. In this case, show

$$P(\mathcal{E}) = \frac{\text{Number of outcomes in } \mathcal{E}}{\text{Total number of outcomes in } \Omega}$$

5.12. You flip a fair coin three times. What is the probability of seeing HTH? (i.e. Heads, then Tails, then Heads)

5.13. You flip a fair coin three times. What is the probability of seeing two heads and one tail?

5.14. You remove the king of hearts from a standard deck of cards, then shuffle it and draw a card.

(a) What is the probability this card is a king?

(b) What is the probability this card is a heart?

5.15. You shuffle a standard deck of cards, then draw four cards.
- **(a)** What is the probability all four are the same suit?
- **(b)** What is the probability all four are red?
- **(c)** What is the probability each has a different suit?

5.16. You roll three fair six-sided dice and add the numbers. What is the probability the result is even?

5.17. You roll three fair six-sided dice and add the numbers. What is the probability the result is even *and* not divisible by 20?

5.18. You shuffle a standard deck of cards, then draw seven cards. What is the probability that you see no aces?

5.19. Show that $P(\mathcal{A} - (\mathcal{B} \cup \mathcal{C})) = P(\mathcal{A}) - P(\mathcal{A} \cap \mathcal{B}) - P(\mathcal{A} \cap \mathcal{C}) + P(\mathcal{A} \cap \mathcal{B} \cap \mathcal{C})$.

5.20. You draw a single card from a standard 52 card deck. What is the probability that it is red?

5.21. You remove all heart cards from a standard 52 card deck, then draw a single card from the result. What is the probability that the card you draw is red?

Conditional Probability

5.22. You roll two fair six-sided dice. What is the conditional probability the sum of numbers is greater than three, conditioned on the first die coming up even.

5.23. You take a standard deck of cards, shuffle it, and remove one card. You then draw a card.
- **(a)** What is the conditional probability that the card you draw is a red king, conditioned on the removed card being a king?
- **(b)** What is the conditional probability that the card you draw is a red king, conditioned on the removed card being a red king?
- **(c)** What is the conditional probability that the card you draw is a red king, conditioned on the removed card being a black ace?

5.24. A royal flush is a hand of five cards, consisting of Ace, King, Queen, Jack and 10 of a single suit. Poker players like this hand, but don't see it all that often.
- **(a)** You draw five cards from a standard deck of playing cards. What is the probability of getting a royal flush?
- **(b)** You draw three cards from a standard deck of playing cards. These are Ace, King, Queen of hearts. What is the probability that the next two cards you draw will result in a getting a royal flush? (this is the conditional probability of getting a royal flush, conditioned on the first three cards being AKQ of hearts).

5.25. You roll a fair five-sided die, and a fair six-sided die.
- **(a)** What is the probability that the sum of numbers is even?
- **(b)** What is the conditional probability that the sum of numbers is even, conditioned on the six-sided die producing an odd number?

Independence

5.26. You take a standard deck of cards, shuffle it, and remove both red kings. You then draw a card.
- **(a)** Is the event {card is red} independent of the event {card is a queen}?
- **(b)** Is the event {card is black} independent of the event {card is a king}?

The Monty Hall Problem

5.27. Monty Hall, Rule 3: If the host uses rule 3, then what is $P(C_1|G_2, r_3)$? Do this by computing conditional probabilities.

5.28. Monty Hall, Rule 4: If the host uses rule 4, and shows you a goat behind door 2, what is $P(C_1|G_2, r_4)$? Do this by computing conditional probabilities.

CHAPTER 6

Random Variables and Expectations

6.1 RANDOM VARIABLES

Quite commonly, we would like to deal with numbers that are random. We can do so by linking numbers to the outcome of an experiment. We define a **random variable**:

Definition: 6.1 *Discrete random variable*

Given a sample space Ω, a set of events $\mathcal{F}$, and a probability function P, and a countable set of of real numbers D, a discrete random variable is a function with domain Ω and range D.

This means that for any outcome ω there is a number $X(\omega)$. P will play an important role, but first we give some examples.

Example: 6.1 *Numbers from coins*

We flip a coin. Whenever the coin comes up heads, we report 1; when it comes up tails, we report 0. This is a random variable.

Example: 6.2 *Numbers from coins II*

We flip a coin 32 times. We record a 1 when it comes up heads, and when it comes up tails, we record a 0. This produces a 32 bit random number, which is a random variable.

Example: 6.3 *The number of pairs in a poker hand*

(from Stirzaker). We draw a hand of five cards. The number of pairs in this hand is a random variable, which takes the values $0, 1, 2$ (depending on which hand we draw)

A function of a discrete random variable is also a discrete random variable.

Example: 6.4 *Parity of coin flips*

We flip a coin 32 times. We record a 1 when it comes up heads, and when it comes up tails, we record a 0. This produces a 32 bit random number, which is a random variable. The parity of this number is also a random variable.

Associated with any value x of the random variable X are a series of events. The most important is the set of outcomes such that $X = x$, which we can write $\{\omega : X(\omega) = x\}$; it is usual to simplify to $\{X = x\}$, and we will do so. The probability that a random variable X takes the value x is given by $P(\{X = x\})$. This is sometimes written as $P(X = x)$, and rather often written as $P(x)$.

We could also be interested in the set of outcomes such that $X \leq x$ (i.e. in $\{\omega : X(\omega) \leq x\}$), which we will write $\{X \leq x\}$; The probability that X takes the value x is given by $P(\{X \leq x\})$. This is sometimes written as $P(X \leq x)$. Similarly, we could be interested in $\{X > x\}$, and so on.

Definition: 6.2 *The probability distribution of a discrete random variable*

The probability distribution of a discrete random variable is the set of numbers $P(\{X = x\})$ for each value x that X can take. The distribution takes the value 0 at all other numbers. Notice that this is non-negative.

Definition: 6.3 *The cumulative distribution of a discrete random variable*

The cumulative distribution of a discrete random variable is the set of numbers $P(\{X <= x\})$ for each value x that X can take. Notice that this is a non-decreasing function of x. Cumulative distributions are often written with an f, so that $f(x)$ might mean $P(\{X <= x\})$.

Worked example 6.1 *Numbers from coins III*

We flip a biased coin 2 times. The flips are independent. The coin has $P(H) = p$, $P(T) = 1 - p$. We record a 1 when it comes up heads, and when it comes up tails, we record a 0. This produces a 2 bit random number, which is a random variable taking the values 0, 1, 2, 3. What is the probability distribution and cumulative distribution of this random variable?

Solution: Probability distribution: $P(0) = (1-p)^2$; $P(1) = (1-p)p$; $P(2) = p(1-p)$; $P(3) = p^2$. Cumulative distribution: $f(0) = (1-p)^2$; $f(1) = (1-p)$; $f(2) = p(1-p) + (1-p) = (1-p^2)$; $f(3) = 1$.

Worked example 6.2 *Betting on coins*

One way to get a random variable is to think about the reward for a bet. We agree to play the following game. I flip a coin. The coin has $P(H) = p$, $P(T) = 1 - p$. If the coin comes up heads, you pay me q; if the coin comes up tails, I pay you r. The number of dollars that change hands is a random variable. What is its probability distribution?

Solution: We see this problem from my perspective. If the coin comes up heads, I get q; if it comes up tails, I get $-r$. So we have $P(X = q) = p$ and $P(X = -r) = (1 - p)$, and all other probabilities are zero.

6.1.1 Joint and Conditional Probability for Random Variables

All the concepts of probability that we described for events carry over to random variables. This is as it should be, because random variables are really just a way of getting numbers out of events. However, terminology and notation change a bit.

Assume we have two random variables X and Y. The probability that X takes the value x and Y takes the value y could be written as $P(\{X = x\} \cap \{Y = y\})$. It is more usual to write it as $P(x, y)$. You can think of this as a table of values, one for each possible pair of x and y values. This table is usually referred to as the **joint probability distribution** of the random variables. Nothing (except notation) has really changed here, but the change of notation is useful.

We will simplify notation further. Usually, we are interested in random variables, rather than potentially arbitrary outcomes or sets of outcomes. We will write $P(X)$ to denote the probability distribution of a random variable, and $P(x)$ or $P(X = x)$ to denote the probability that that random variable takes a particular value. This means that, for example, the rule we could write as

$$P(\{X = x\} \,|\, \{Y = y\})P(\{Y = y\}) = P(\{X = x\} \cap \{Y = y\})$$

will be written as

$$P(x|y)P(y) = P(x, y).$$

This yields **Bayes' rule**, which is important enough to appear in its own box.

Definition: 6.4 *Bayes' rule*

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)}$$

Random variables have another useful property. If $x_0 \neq x_1$, then the event $\{X = x_0\}$ must be disjoint from the event $\{X = x_1\}$. This means that

$$\sum_x P(x) = 1$$

and that, for any y,

$$\sum_x P(x|y) = 1$$

(if you're uncertain on either of these points, check them by writing them out in the language of events).

Now assume we have the joint probability distribution of two random variables, X and Y. Recall that we write $P(\{X = x\} \cap \{Y = y\})$ as $P(x, y)$. Now consider the sets of outcomes $\{Y = y\}$ for each different value of y. These sets must be disjoint, because y cannot take two values at the same time. Furthermore, each element of the set of outcomes $\{X = x\}$ must lie in one of the sets $\{Y = y\}$. So we have

$$\sum_y P(\{X = x\} \cap \{Y = y\}) = P(\{X = x\})$$

which is usually written as

$$\sum_y P(x, y) = P(x)$$

and is often referred to as the **marginal probability** of X.

Definition: 6.5 *Independent random variables*

The random variables X and Y are **independent** if the events $\{X = x\}$ and $\{Y = y\}$ are independent. This means that

$$P(\{X = x\} \cap \{Y = y\}) = P(\{X = x\})P(\{Y = y\}),$$

which we can rewrite as

$$P(x, y) = P(x)P(y)$$

Worked example 6.3 *Sums and differences of dice*

You throw two dice. The number of spots on the first die is a random variable (call it X); so is the number of spots on the second die (Y). Now define $S = X + Y$ and $D = X - Y$. What is the probability distribution of S and of D?

Solution: S can have values in the range $2, \ldots, 12$. There is only one way to get a $S = 2$; two ways to get $S = 3$; and so on. Using the methods of chapter 1 for each case, the probabilities for $[2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]$ are $[1, 2, 3, 4, 5, 6, 5, 4, 3, 2, 1]/36$. Similarly, D can have values in the range $-5, \ldots, 5$. Again, using the methods of chapter 1, the probabilities for $[-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5]$ are $[1, 2, 3, 4, 5, 6, 5, 4, 3, 2, 1]/36$.

Worked example 6.4 *Sums and differences of dice, II*

Using the terminology of example 3, what is the joint probability distribution of S and D?

Solution: This is more interesting to display, because it's an 11x11 table. Each entry of the table represents a pair of S, D values. Many pairs can't occur (for example, for $S = 2$, D can only be zero; if S is even, then D must be even; and so on). You can work out the table by checking each case; it's in Table 6.1.

Worked example 6.5 *Sums and differences of dice, III*

Using the terminology of example 3, are X and Y independent? are S and D independent?

Solution: X and Y are clearly independent. But S and D are not. There are several ways to see this. One way is to notice that, if you know $S = 2$, then you know the value of D precisely; but if you know $S = 3$, D could be either 1 or -1. This means that $P(S|D)$ depends on D, so they're not independent. Another way is to notice that the rank of the table, as a matrix, is 6, which means that it can't be the outer product of two vectors.

$$\frac{1}{36} \times \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

TABLE 6.1: *A table of the joint probability distribution of S (vertical axis; scale $2, \ldots, 12$) and D (horizontal axis; scale $-5, \ldots, 5$) from example 4*

Worked example 6.6 *Sums and differences of dice, IV*

Using the terminology of example 3, what is $P(S|D = 0)$? what is $P(D|S = 11)$?

Solution: You could work it out either of these from the table, or by first principles. If $D = 0$, S can have values $2, 4, 6, 8, 10, 12$, and each value has conditional probability $1/6$. If $S = 11$, D can have values 1, or -1, and each value has conditional probability $1/2$.

6.1.2 Just a Little Continuous Probability

Our random variables take values from a discrete set of numbers D. This makes the underlying machinery somewhat simpler to describe, and is often, but not always, enough for model building. Some phenomena are more naturally modelled as being continuous — for example, human height; human weight; the mass of a distant star; and so on. Giving a complete formal description of probability on a continuous space is surprisingly tricky, and would involve us in issues that do not arise much in practice.

These issues are caused by two interrelated facts: real numbers have infinite precision; and you can't count real numbers. A continuous random variable is still a random variable, and comes with all the stuff that a random variable comes with. We will not speculate on what the underlying sample space is, nor on the underlying events. This can all be sorted out, but requires moderately heavy lifting that isn't particularly illuminating for us. The most interesting thing for us is specifying the probability distribution. Rather than talk about the probability that a real number takes a particular value (which we can't really do satisfactorily most of the time), we will instead talk about the probability that it lies in some interval. So we can specify a probability distribution for a continuous random variable by giving a set

of (very small) intervals, and for each interval providing the probability that the random variable lies in this interval.

The easiest way to do this is to supply a **probability density function**. Let $p(x)$ be a probability density function for a continuous random variable X. We interpret this function by thinking in terms of small intervals. Assume that dx is an infinitesimally small interval. Then

$$p(x)dx = P(\{\text{event that } X \text{ takes a value in the range } [x, x + dx]\}).$$

Important properties of probability density functions follow from this definition.

Useful Facts: 6.1 *Probability density functions*

- Probability density functions are non-negative. This follows from the definition; a negative value at some x would imply a negative probability.
- For $a < b$

  $$P(\{\text{event that } X \text{ takes a value in the range } [a, b]\}) = \int_a^b p(x)dx.$$

  which we obtain by summing $p(x)dx$ over all the infinitesimal intervals between a and b.
- We must have that

  $$\int_{-\infty}^{\infty} p(x)dx = 1.$$

  This is because

  $$P(\{X \text{ takes a value in the range } [-\infty, \infty]\}) = 1 = \int_{-\infty}^{\infty} p(x)dx$$

- Probability density functions are usually called pdf's.
- It is quite usual to write all pdf's as lower-case p's. If one specifically wishes to refer to probability, one writes an upper case P, as in the previous points.

One good way to think about pdf's is as the limit of a histogram. Imagine you collect an arbitrarily large dataset of data items, each of which is independent. You build a histogram of that dataset, using arbitrarily narrow boxes. You scale the histogram so that the sum of the box areas is one. The result is a probability density function.

The pdf doesn't represent the probability that a random variable takes a value. Instead, you should think of $p(x)$ as being the limit of a ratio (which is why

it's called a density):

$$\frac{\text{the probability that the random variable will lie in a small interval centered on } x}{\text{the length of the small interval centered on } x}$$

Notice that, while a pdf has to be non-negative, and it has to integrate to 1, it does *not* have to be smaller than one. A ratio like this could be a lot larger than one, as long as it isn't larger than one for too many x (because the integral must be one).

Probability density functions can be moderately strange functions.

Worked example 6.7 *Strange probability density functions*

There is some (small!) voltage over the terminals of a warm resistor caused by noise (electrons moving around in the heat and banging into one another). This is a good example of a continuous random variable, and we can assume there is some probability density function for it, say $p(x)$. We assume that $p(x)$ has the property that

$$\lim_{\epsilon \to 0} \int_{v-\epsilon}^{v+\epsilon} p(x)dx = 0$$

which is what you'd expect for any function you're likely to have dealt with. Now imagine I define a new random variable by the following procedure: I flip a coin; if it comes up heads, I report 0; if tails, I report the voltage over the resistor. This random variable, u, has a probability $1/2$ of taking the value 0, and $1/2$ of taking a value from $p(x)$. Write this random variable's probability density function $q(u)$. Compute

$$\lim_{\epsilon \to 0} \int_{-\epsilon}^{\epsilon} q(u)du$$

Solution: We can do this from the definition. We have

$$P(\{u \in [-\epsilon, \epsilon]\}) = \int_{-\epsilon}^{\epsilon} q(u)du.$$

But u will take the value 0 with probability $1/2$, and otherwise takes the value over the resistor. So

$$P(\{u \in [-\epsilon, \epsilon]\}) = 1/2 + \int_{-\epsilon}^{\epsilon} p(x)dx.$$

and

$$\lim_{\epsilon \to 0} \int_{-\epsilon}^{\epsilon} q(u)du = \lim_{\epsilon \to 0} P(\{u \in [-\epsilon, \epsilon]\}) = 1/2.$$

This means $q(x)$ has the property that

$$\lim_{\epsilon \to 0} \int_{-\epsilon}^{\epsilon} q(u)du = 1/2,$$

which means that $q(u)$ is displaying quite unusual behavior at $u = 0$. We will not need to deal with probability density functions that behave like this; but you should be aware of the possibility.

Every probability density function $p(x)$ has the property that $\int_{-\infty}^{\infty} p(x)dx = 1$; this is useful, because when we are trying to determine a probability density

function, we can ignore a constant factor. So if $g(x)$ is a non-negative function that is proportional to the probability density function (often pdf) we are interested in, we can recover the pdf by computing

$$p(x) = \frac{1}{\int_{-\infty}^{\infty} g(x)dx} g(x).$$

This procedure is sometimes known as **normalizing**, and $\int_{-\infty}^{\infty} g(x)dx$ is the **normalizing constant**.

6.2 EXPECTATIONS AND EXPECTED VALUES

Imagine we play the game of example 2 multiple times. Our frequency definition of probability means that in N games, we expect to see about pN heads and $(1-p)N$ tails. In turn, this means that my total income from these N games should be about $(pN)q - ((1-p)N)r$. The N in this expression is inconvenient; instead, we could say that for any single game, my income is

$$pq - (1-p)r.$$

This isn't the actual income from a single game (which would be either q or $-r$, depending on what the coin did). Instead, it's an estimate of what would happen over a large number of games, on a per-game basis. This is an example of an **expected value**.

6.2.1 Expected Values of Discrete Random Variables

Definition: 6.6 *Expected value*

Given a discrete random variable X which takes values in the set $\mathcal{D}$ and which has probability distribution P, we define the expected value

$$\mathbb{E}[X] = \sum_{x \in \mathcal{D}} xP(X = x).$$

This is sometimes written which is $\mathbb{E}_P[X]$, to clarify which distribution one has in mind

Notice that an expected value could take a value that the random variable doesn't take.

Example: 6.5 *Betting on coins*

We agree to play the following game. I flip a fair coin (i.e. $P(H) = P(T) = 1/2$). If the coin comes up heads, you pay me 1; if the coin comes up tails, I pay you 1. The expected value of my income is 0, even though the random variable never takes that value.

Definition: 6.7 *Expectation*

Assume we have a function f that maps a discrete random variable X into a set of numbers $\mathcal{D}_f$. Then $f(x)$ is a discrete random variable, too, which we write F. The expected value of this random variable is written

$$\mathbb{E}[f] = \sum_{u \in \mathcal{D}_f} uP(F = u) = \sum_{x \in \mathcal{D}} f(x)P(X = x)$$

which is sometimes referred to as "the expectation of f". The process of computing an expected value is sometimes referred to as "taking expectations".

Expectations are linear, so that $\mathbb{E}[0] = 0$ and $\mathbb{E}[A + B] = \mathbb{E}[A] + \mathbb{E}[B]$. The expectation of a constant is that constant (or, in notation, $\mathbb{E}[k] = k$), because probabilities sum to 1. Because probabilities are non-negative, the expectation of a non-negative random variable must be non-negative.

6.2.2 Expected Values of Continuous Random Variables

We can compute expectations for continuous random variables, too, though summing over all values now turns into an integral. This should be expected. Imagine you choose a set of closely spaced values for x which are x_i, and then think about x as a discrete random variable. The values are separated by steps of width Δx. Then the expected value of this discrete random variable is

$$\mathbb{E}[X] = \sum_i x_i P(X \in \text{interval centered on } x_i) = \sum_i x_i p(x_i)\Delta x$$

and, as the values get closer together and Δx gets smaller, the sum limits to an integral.

Definition: 6.8 *Expected value*

Given a continuous random variable X which takes values in the set $\mathcal{D}$ and which has probability distribution P, we define the expected value

$$\mathbb{E}[X] = \int_{x \in \mathcal{D}} xp(x)dx.$$

This is sometimes written $\mathbb{E}_p[X]$, to clarify which distribution one has in mind

The expected value of a continuous random variable could be a value that the random variable doesn't take, too. Notice one attractive feature of the $\mathbb{E}[X]$ notation; we don't need to make any commitment to whether X is a discrete random variable (where we would write a sum) or a continuous random variable (where we would write an integral). The reasoning by which we turned a sum into an integral works for functions of continuous random variables, too.

Definition: 6.9 *Expectation*

Assume we have a function f that maps a continuous random variable X into a set of numbers $\mathcal{D}_f$. Then $f(x)$ is a continuous random variable, too, which we write F. The expected value of this random variable is

$$\mathbb{E}[f] = \int_{x \in \mathcal{D}} f(x)p(x)dx$$

which is sometimes referred to as "the expectation of f". The process of computing an expected value is sometimes referred to as "taking expectations".

Again, for continuous random variables, expectations are linear, so that $\mathbb{E}[0] = 0$ and $\mathbb{E}[A + B] = \mathbb{E}[A]+\mathbb{E}[B]$. The expectation of a constant is that constant (or, in notation, $\mathbb{E}[k] = k$), because probabilities sum to 1. Because probabilities are non-negative, the expectation of a non-negative random variable must be non-negative.

6.2.3 Mean, Variance and Covariance

There are three very important expectations with special names.

Definition: 6.10 *The mean or expected value*

The mean or expected value of a random variable X is

$$\mathbb{E}[X]$$

Definition: 6.11 *The variance*

The variance of a random variable X is

$$\mathsf{var}[X] = \mathbb{E}\big[(X - \mathbb{E}[X])^2\big]$$

Notice that

$$\begin{aligned}\mathbb{E}\big[(X - \mathbb{E}[X])^2\big] &= \mathbb{E}\Big[\big(X^2 - 2X\mathbb{E}[X] + \mathbb{E}[X]^2\big)\Big] \\ &= \mathbb{E}\big[X^2\big] - 2\mathbb{E}[X]\mathbb{E}[X] + \mathbb{E}[X]^2 \\ &= \mathbb{E}\big[X^2\big] - (\mathbb{E}[X])^2\end{aligned}$$

Definition: 6.12 *The covariance*

The covariance of two random variables X and Y is

$$\mathsf{cov}\,(X, Y) = \mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])]$$

Notice that

$$\begin{aligned}\mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])] &= \mathbb{E}[(XY - Y\mathbb{E}[X] - X\mathbb{E}[Y] + \mathbb{E}[X]\mathbb{E}[Y])] \\ &= \mathbb{E}[XY] - 2\mathbb{E}[Y]\mathbb{E}[X] + \mathbb{E}[X]\mathbb{E}[Y] \\ &= \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y].\end{aligned}$$

We also have $\mathsf{var}[X] = \mathsf{cov}\,(X, X)$.

Now assume that we have a probability distribution $P(X)$ defined on some discrete set of numbers. There is some random variable that produced this probability distribution. This means that we could talk about the mean of a probability

distribution P (rather than the mean of a random variable whose probability distribution is $P(X)$). It is quite usual to talk about the mean of a probability distribution. Furthermore, we could talk about the variance of a probability distribution P (rather than the variance of a random variable whose probability distribution is $P(X)$).

Worked example 6.8 *Variance*

Can a random variable have $\mathbb{E}[X] > \sqrt{\mathbb{E}[X^2]}$?

Solution: No, because that would mean that $\mathbb{E}\left[(X - \mathbb{E}[X])^2\right] < 0$. But this is the expected value of a non-negative quantity; it must be non-negative.

Worked example 6.9 *More variance*

We just saw that a random variable can't have $\mathbb{E}[X] > \sqrt{\mathbb{E}[X^2]}$. But I can easily have a random variable with large mean and small variance - isn't this a contradiction?

Solution: No, you're confused. Your question means you think that the variance of X is given by $\mathbb{E}\left[X^2\right]$; but actually $\mathsf{var}[X] = \mathbb{E}\left[X^2\right] - \mathbb{E}[X]^2$

Worked example 6.10 *Mean of a coin flip*

We flip a biased coin, with $P(H) = p$. The random variable X has value 1 if the coin comes up heads, 0 otherwise. What is the mean of X? (i.e. $\mathbb{E}[X]$).

Solution: $\mathbb{E}[X] = \sum_{x \in D} xP(X = x) = 1p + 0(1 - p) = p$

Useful Facts: 6.2 *Expectations*

1. $\mathbb{E}[0] = 0$
2. $\mathbb{E}[X + Y] = \mathbb{E}[X] + \mathbb{E}[Y]$
3. $\mathbb{E}[kX] = k\mathbb{E}[X]$
4. $\mathbb{E}[1] = 1$
5. if X and Y are independent, then $\mathbb{E}[XY] = \mathbb{E}[X]\mathbb{E}[Y]$.
6. if X and Y are independent, then $\mathsf{cov}(X, Y) = 0$.

All but 5 and 6 are obvious from the definition. If 5 is true, then 6 is obviously true. I prove 5.

Proposition: *If X and Y are independent random variables, then $\mathbb{E}[XY] = \mathbb{E}[X]\mathbb{E}[Y]$.*

Proof: Recall that $\mathbb{E}[X] = \sum_{x \in D} xP(X = x)$, so that

$$\begin{aligned}
\mathbb{E}[XY] &= \sum_{(x,y) \in D_x \times D_y} xyP(X = x, Y = y) \\
&= \sum_{x \in D_x} \sum_{y \in D_y} (xyP(X = x, Y = y)) \\
&= \sum_{x \in D_x} \sum_{y \in D_y} (xyP(X = x)P(Y = y)) \\
&\quad \text{because } X \text{ and } Y \text{ are independent} \\
&= \sum_{x \in D_x} \sum_{y \in D_y} (xP(X = x))\,(yP(Y = y)) \\
&= \left(\sum_{x \in D_x} xP(X = x) \right) \left(\sum_{y \in D_y} yP(Y = y) \right) \\
&= (\mathbb{E}[X])(\mathbb{E}[Y]).
\end{aligned}$$

This is certainly not true when X and Y are not independent (try $Y = -X$).

Useful Facts: 6.3 *Variance*

It is quite usual to write $\mathsf{var}[X]$ for the variance of the random variable X.

1. $\mathsf{var}[0] = 0$
2. $\mathsf{var}[1] = 0$
3. $\mathsf{var}[X] \geq 0$
4. $\mathsf{var}[kX] = k^2\mathsf{var}[X]$
5. if X and Y are independent, then $\mathsf{var}[X+Y] = \mathsf{var}[X] + \mathsf{var}[Y]$

1, 2, 3 are obvious. You will prove 4 and 5 in the exercises.

Worked example 6.11 *Variance of a coin flip*

We flip a biased coin, with $P(H) = p$. The random variable X has value 1 if the coin comes up heads, 0 otherwise. What is the variance of X? (i.e. $\mathsf{var}[X]$).

Solution: $\mathsf{var}[X] = \mathbb{E}\left[(X - \mathbb{E}[X])^2\right] = \mathbb{E}\left[X^2\right] - \mathbb{E}[X]^2 = (1p - 0(1-p)) - p^2 = p(1-p)$

The variance of a random variable is often inconvenient, because its units are the square of the units of the random variable. Instead, we could use the **standard deviation**.

Definition: 6.13 *Standard deviation*

The **standard deviation** of a random variable X is defined as

$$\mathsf{std}(X) = \sqrt{\mathsf{var}[X]}$$

You do need to be careful with standard deviations. If X and Y are independent random variables, then $\mathsf{var}[X+Y] = \mathsf{var}[X] + \mathsf{var}[Y]$, but $\mathsf{std}(X+Y) = \sqrt{\mathsf{std}(X)^2 + \mathsf{std}(Y)^2}$. One way to avoid getting mixed up is to remember that variances add, and derive expressions for standard deviations from that.

6.2.4 Expectations and Statistics

You should have noticed we now have two notions each for mean, variance, covariance, and standard deviation. One, which we expounded in sections 1, describes

datasets. We will call these **descriptive statistics**. The other, described above, is a property of probability distributions. We will call these **expectations**. In each case, the reason we have one name for two notions is that the notions are not really all that different.

Imagine we have a dataset $\{\mathbf{x}\}$ of N items, where the i'th item is $\mathbf{x}_i$. We can build a probability distribution out of this dataset, by placing a probability on each data item. We will give each data item the same probability (which must be $1/N$, so all probabilities add to 1). Write $\mathbb{E}[\mathbf{x}]$ for the mean of this distribution. We have

$$\mathbb{E}[\mathbf{x}] = \sum_i \mathbf{x}_i p(\mathbf{x}_i) = \frac{1}{N} \sum_i \mathbf{x}_i = \text{mean}\left(\{\mathbf{x}\}\right).$$

The variances, standard deviations and covariance have the same property: For this particular distribution (sometimes called the **empirical distribution**), the expectations have the same value as the descriptive statistics (exercises).

In section 1, we will see a form of converse to this fact. Imagine we have a dataset that consists of independent, identically distributed samples from a probability distribution. That is, we know that each data item was obtained independently from the distribution. For example, we might have a count of heads in each of a number of coin flip experiments. Then the descriptive statistics will turn out to be accurate estimates of the expectations.

6.2.5 Indicator Functions

It is sometimes convenient when working with random variables to use **indicator functions**. This is a function that is one when some condition is true, and zero otherwise. The reason they are useful is that their expectated values have interesting properties.

Definition: 6.14 *Indicator functions*

An indicator function for an event is a function that takes the value zero for values of X where the event does not occur, and one where the event occurs. For the event $\mathcal{E}$, we write

$$\mathbb{I}_{[\mathcal{E}]}(X)$$

for the relevant indicator function.

For example,

$$\mathbb{I}_{[\{|X|\}\leq a]}(X) = \begin{cases} 1 & \text{if } -a < X < a \\ 0 & \text{otherwise} \end{cases}$$

Indicator functions have one useful property.

$$\mathbb{E}\left[\mathbb{I}_{[\mathcal{E}]}\right] = P(\mathcal{E})$$

which you can establish by checking the definition of expectations.

6.2.6 Two Inequalities

Mean and variance tell us quite a lot about a random variable, as two important inequalities show.

Definition: 6.15 *Markov's inequality*

Markov's inequality is

$$P(\{\| X \| \geq a\}) \leq \frac{\mathbb{E}[\| X \|]}{a}.$$

You should read this as indicating that a random variable is most unlikely to have an absolute value a lot larger than the mean of its absolute value. This should seem fairly intuitive from the definition of expectation. Recall that

$$\mathbb{E}[X] = \sum_{x \in D} xP(\{X = x\})$$

Assume that D contains only non-negative numbers (that absolute value). Then the only way to have a small value of $\mathbb{E}[X]$ is to be sure that, when x is large, $P(\{X = x\})$ is small. The proof is a rather more formal version of this observation, below.

Definition: 6.16 *Chebyshev's inequality*

Chebyshev's inequality is

$$P(\{|X - \mathbb{E}[X]| \geq a\}) \leq \frac{\mathsf{var}[X]}{a^2}.$$

It is common to see this in another form, obtained by writing σ for the standard deviation of X, substituting $k\sigma$ for a, and rearranging

$$P(\{|X - \mathbb{E}[X]| \geq k\sigma\}) \leq \frac{1}{k^2}$$

This means that the probability of a random variable taking a particular value must fall off rather fast as that value moves away from the mean, in units scaled to the variance. This probably doesn't seem intuitive from the definition of expectation. But think about it this way: values of a random variable that are many standard deviations above the mean must have low probability, otherwise the standard deviation would be bigger. The proof, again, is a rather more formal version of this observation, and appears below.

Proposition: *Markov's inequality*

$$P(\{\| X \| \geq a\}) \leq \frac{\mathbb{E}[\| X \|]}{a}.$$

Proof: (from Wikipedia). Notice that, for $a > 0$,

$$a\mathbb{I}_{[\{|X| \geq a\}]}(X) \leq |X|$$

(because if $|X| \geq a$, the LHS is a; otherwise it is zero). Now we have

$$\mathbb{E}\left[a\mathbb{I}_{[\{|X| \geq a\}]}\right] \leq \mathbb{E}[|X|]$$

but, because expectations are linear, we have

$$\mathbb{E}\left[a\mathbb{I}_{[\{|X| \geq a\}]}\right] = a\mathbb{E}\left[\mathbb{I}_{[\{|X| \geq a\}]}\right] = aP(\{|X| \geq a\})$$

and so we have

$$aP(\{|X| \geq a\}) \leq \mathbb{E}[|X|]$$

and we get the inequality by division, which we can do because $a > 0$.

Proposition: *Chebyshev's inequality*

$$P(\{|X - \mathbb{E}[X]| \geq a\}) \leq \frac{\mathsf{var}[X]}{a^2}.$$

Proof: Write U for the random variable $(X - \mathbb{E}[X])^2$. Markov's inequality gives us

$$P(\{|U| \geq w\}) \leq \frac{\mathbb{E}[|U|]}{w}$$

Now notice that, if $a^2 = w$,

$$P(\{|U| \geq w\}) = P(\{|X - \mathbb{E}[X]|\} \geq a)$$

so we have

$$P(\{|U| \geq w\}) = P(\{|X - \mathbb{E}[X]|\} \geq a) \leq \frac{\mathbb{E}[|U|]}{w} = \frac{\mathsf{var}[X]}{a^2}$$

6.2.7 IID Samples and the Weak Law of Large Numbers

Imagine a random variable X, obtained by flipping a fair coin and reporting 1 for an H and -1 for a T. We can talk about the probability distribution $P(X)$ of this random variable; we can talk about the expected value that the random variable takes; but the random variable itself doesn't have a value. However, if we actually flip a coin, we get either a 1 or a -1. This number is often called a **sample** of the random variable (or of its probability distribution). Similarly, if we flipped a coin many times, we'd have a set of numbers (or samples). These numbers would be independent. Their histogram would look like $P(X)$. Collections of data items like this are important enough to have their own name.

Assume we have a set of data items x_i such that (a) they are independent; (b) each is produced from the same process; and (c) the histogram of a very large set of data items looks increasingly like the probability distribution $P(X)$ as the number of data items increases. Then we refer to these data items as **independent identically distributed samples** of $P(X)$; for short, **iid samples** or even just **samples**. For all of the cases we will deal with, it will be obvious how to get IID samples. However, it's worth knowing that obtaining IID samples from arbitrary probability distributions is very difficult.

Now assume we have a set of N IID samples x_i of a probability distribution $P(X)$. Write

$$X_N = \frac{\sum_{i=1}^{N} x_i}{N}.$$

Now X_N is a random variable (the x_i are IID samples, and for a different set of samples you will get a different, random, X_N). Notice that $P(X = x_1, X = x_2, \ldots, X = x_n) = P(X = x_1)P(X = x_2)\ldots P(X = x_n)$, because the samples are independent and each is a sample of $P(X)$. Now

$$\mathbb{E}[X_N] = \mathbb{E}[X]$$

because

$$\mathbb{E}[X_N] = \left(\frac{1}{N}\right) \sum_{i=1}^{N} \mathbb{E}[X].$$

This means that

$$\frac{\sum_{i=1}^{N} x_i}{N}$$

should be give an accurate estimate of $\mathbb{E}[X]$. In fact, as N gets large, the estimate becomes more accurate.

Definition: 6.17 *The Weak Law of Large Numbers*

The **weak law of large numbers** states that, if $P(X)$ has finite variance, then for any positive number ϵ

$$\lim_{N\to\infty} P(\{\| X_N - \mathbb{E}[X] \| \geq \epsilon\}) = 0.$$

Equivalently, we have

$$\lim_{N\to\infty} P(\{\| X_N - \mathbb{E}[X] \| < \epsilon\}) = 1.$$

Each form means that, for a large enough set of IID samples, the average of the samples (i.e. X_N) will, with high probability, be very close to the expectation $\mathbb{E}[X]$.

Proposition: *Weak law of large numbers*

$$\lim_{N\to\infty} P(\{\|X_N - \mathbb{E}[X]\| \geq \epsilon\}) = 0.$$

Proof: Assume that $P(X)$ has finite variance; that is, $\mathsf{var}(\{X\}) = \sigma^2$. Choose $\epsilon > 0$. Now we have that

$$\begin{aligned}
\mathsf{var}(\{X_N\}) &= \mathsf{var}\left(\left\{\frac{\sum_{i=1} Nx_i}{N}\right\}\right) \\
&= (\frac{1}{N^2})\mathsf{var}\left(\left\{\sum_{i=1} Nx_i\right\}\right) \\
&= (\frac{1}{N^2})(N\sigma^2) \\
&\quad \text{because the } x_i \text{ are independent} \\
&= \frac{\sigma^2}{N}
\end{aligned}$$

and that

$$\mathbb{E}[X_N] = \mathbb{E}[X].$$

Now Chebyshev's inequality gives

$$P(\{\|X_N - \mathbb{E}[X]\| \geq \epsilon\}) \leq \frac{\sigma^2}{N\epsilon^2}$$

so

$$\lim_{N\to\infty} P(\{\|X_N - \mathbb{E}[X]\| \geq \epsilon\}) = \lim_{N\to\infty} \frac{\sigma^2}{N\epsilon^2} = 0.$$

Notice that

$$1 - P(\{\|X_N - \mathbb{E}[X]\| \geq \epsilon\}) = P(\{\|X_N - \mathbb{E}[X]\| < \epsilon\}) \geq 1 - \frac{\sigma^2}{N\epsilon^2}.$$

so that

$$\lim_{N\to\infty} P(\{\|X_N - \mathbb{E}[X]\| < \epsilon\}) = \lim_{N\to\infty} \left(1 - \frac{\sigma^2}{N\epsilon^2}\right) = 1.$$

6.3 USING EXPECTATIONS

The weak law of large numbers gives us a very valuable way of thinking about expectations. Assume we have a random variable X. Then the weak law says that, if you observe this random variable over a large number of trials, the mean value

you observe should be very close to $\mathbb{E}[X]$. Notice that this extends to functions of random variables (because they are random variables, too). For example, I observe values x_i of a random variable X over a large number N of trials, and compute

$$\frac{1}{N}\sum_{i=1} Nf(x_i).$$

The weak law says that the value I get should be very close to $\mathbb{E}[f]$. You can show this by defining a new random variable $F = f(X)$. This has a probability distribution $P(F)$, which might be difficult to know — but we don't need to. $\mathbb{E}[f]$, the expected value of the function f under the distribution $P(X)$. This is the same as $\mathbb{E}[F]$, and the weak law applies.

Remember: the average over repeated trials of a random variable is very close to the expectation. You can use this information to make many kinds of decision in uncertain circumstances.

6.3.1 Should you accept a bet?

We can't answer this as a moral question, but we can as a practical question, using expectations. Generally, a bet involves an agreement that amounts of money will change hands, depending on the outcome of an experiment. Mostly, you are interested in how much you get from the bet, so it is natural to give sums of money you receive a positive sign, and sums of money you pay out a negative sign. Under this convention, the practical answer is easy: accept a bet enthusiastically if its expected value is positive, otherwise decline it. It is interesting to notice how poorly this advice describes actual human behavior.

Worked example 6.12 *Red or Black?*

A roulette wheel has 36 numbers, 18 of which are red and 18 of which are black. Different wheels also have one, two, or even three zeros,which are colorless. A ball is thrown at the wheel when it is spinning, and it falls into a hole corresponding to one of the numbers (when the number is said to "come up"). The wheel is set up so that there is the same probability of each number coming up. You can bet on (among other things) whether a red number or a black number comes up. If you bet 1 on red, and a red number comes up, you keep your stake and get 1, otherwise you get -1 (i.e. the house keeps your bet).

- On a wheel with one zero, what is the expected value of a 1 bet on red?
- On a wheel with two zeros, what is the expected value of a 1 bet on red?
- On a wheel with three zeros, what is the expected value of a 1 bet on red?

Solution: Write p_r for the probability a red number comes up. The expected value is $1 \times p_r + (-1)(1 - p_r)$ which is $2p_r - 1$.

- In this case, $p_r =$ (number of red numbers)/(total number of numbers) $=$ $18/37$. So the expected value is $-1/37$ (you lose about 3 cents each time you bet).
- In this case, $p_r = 18/38$. So the expected value is $-2/38 = -1/19$ (you lose slightly more than five cents each time you bet).
- In this case, $p_r = 18/39$. So the expected value is $-3/39 = -1/13$ (you lose slightly less than 8 cents each time you bet).

Notice that in the roulette game, the money you lose will go to the house. So the expected value to the house is just the negative of the expected value to you. This is positive, which is a partial explanation of why there are lots of roulette wheels, and usually free food nearby. Not all bets are like this, though.

Worked example 6.13 *Coin game*

In this game, P1 flips a fair coin and P2 calls "H" or "T". If P2 calls right, then P1 throws the coin into the river; otherwise, P1 keeps the coin. What is the expected value of this game to P2? and to P1?

Solution: To P2, which we do first, because it's easiest: P2 gets 0 if P2 calls right, and 0 if P2 calls wrong; these are the only cases, so the expected value is 0. To P1: P1 gets -1 if P2 calls right, and 0 if P1 calls wrong. The coin is fair, so the probability P2 calls right is $1/2$. The expected value is $-1/2$. While I can't explain why people would play such a game, I've actually seen this done.

We call a bet fair when its expected value is zero. Taking a bet with a negative expected value is unwise, because, on average, you will lose money. Worse, the more times you play, the more you lose. Taking a bet with a positive expected value is likely to be profitable. However, you do need to be careful you computed the expected value right.

Worked example 6.14 *Birthdays in succession*

P1 and P2 agree to the following bet. P1 gives P2 a stake of 1. If three people, stopped at random on the street, have birthdays in succession (i.e. Mon-Tue-Wed, and so on), then P2 gives P1 100. Otherwise, P1 loses the stake. What is the expected value of this bet to P1?

Solution: Write p for the probability of winning. Then the expected value is $p \times 100 - (1-p) \times 1$. We computed p in example 1 (it was $1/49$). So the bet is worth $(52/49)$, or slightly more than a dollar, to P1. P1 should be happy to agree to this as often as possible.

The reason P2 agrees to bets like that of example 14 is most likely that P2 can't compute the probability exactly. P2 thinks the event is quite unlikely, so the expected value is negative; but it isn't as unlikely as P2 thought it was, and this is how P1 makes a profit. This is one of the reasons you should be careful accepting a bet from a stranger: they might be able to compute better than you.

6.3.2 Odds, Expectations and Bookmaking — a Cultural Diversion

Gamblers sometimes use a terminology that is a bit different from ours. In particular, the term **odds** is important. The term comes from the following idea: P1 pays a bookmaker b (the stake) to make a bet; if the bet is successful, P1 receives a, and if not, loses the original stake.

Assume the bet is fair, so that the expected value is zero. Write p for the probability of winning. The net income to P1 is $ap - b(1-p)$. If this is zero, then $p = b/(a+b)$. So you can interpret odds in terms of probability, *if* you assume the

bet is fair.

A bookmaker sets adds at which to accept bets from gamblers. The bookmaker does not wish to lose money at this business, and so must set odds which are potentially profitable. Doing so is not simple (bookmakers can, and occasionally do, lose catastrophically, and go out of business). In the simplest case, assume that the bookmaker knows the probability p that a particular bet will win. Then the bookmaker could set odds of $(1-p)/p : 1$. In this case, the expected value of the bet is zero; this is fair, but not attractive business, so the bookmaker will set odds assuming that the probability is a bit higher than it really is. There are other bookmakers out there, so there is some reason for the bookmaker to try to set odds that are close to fair.

In some cases, you can tell when you are dealing with a bookmaker who is likely to go out of business soon. For example, imagine there are two horses running in a race, both at $10 : 1$ odds — whatever happens, you could win by betting 1 on each. There is a more general version of this phenomenon. Assume the bet is placed on a horse race, and that bets pay off only for the winning horse. Assume also that exactly one horse will win (i.e. the race is never scratched, there aren't any ties, etc.), and write the probability that the i'th horse will win as p_i. Then $\sum_{i \in \text{horses}} p_i$ must be 1. Now if the bookmaker's odds yield a set of probabilities that is less than 1, their business should fail, because there is at least one horse on which they are paying out too much. Bookmakers deal with this possibility by writing odds so that $\sum_{i \in \text{horses}} p_i$ is larger than one.

But this is not the only problem a bookmaker must deal with. The bookmaker doesn't actually know the probability that a particular horse will win, and must account for errors in this estimate. One way to do so is to collect as much information as possible (talk to grooms, jockeys, etc.). Another is to look at the pattern of bets that have been placed already. If the bookmaker and the gamblers agree on the probability that each horse will win, then there should be no expected advantage to choosing one horse over another — each should pay out slightly less than zero to the gambler (otherwise the bookmaker doesn't eat). But if the bookmaker has underestimated the probability that a particular horse will win, a gambler may get a positive expected payout by betting on that horse. This means that if one particular horse attracts a lot of money from bettors, it is wise for the bookmaker to offer less generous odds on that horse. There are two reasons: first, the bettors might know something the bookmaker doesn't, and they're signalling it; second, if the bets on this horse are very large and it wins, the bookmaker may not have enough capital left to pay out or to stay in business. All this means that real bookmaking is a complex, skilled business.

6.3.3 Ending a Game Early

Imagine two people are playing a game for a stake, but must stop early — who should get what percentage of the stake? One way to do this is to give each player what they put in at the start, but this is (mildly) unfair if one has an advantage over the other. The alternative is to give each player the expected value of the game at that state for that player. Sometimes one can compute that expectation quite easily.

Worked example 6.15 *Ending a game early*

(from Durrett), two players each pay 25 to play the following game. They toss a fair coin. If it comes up heads, player H wins that toss; if tails, player T wins. The first player to reach 10 wins takes the stake of 50. But one player is called away when the state is 8-7 (H-T) — how should the stake be divided?

Solution: In this state, each player can either win — and so get 50 — or lose — and so get 0. The expectation for H is $50P(\{\text{H wins from 8-7}\}) + 0P(\{\text{T wins from 8-7}\})$, so we need to compute $P(\{\text{H wins from 8-7}\})$. Similarly, the expectation for T is $50P(\{\text{T wins from 8-7}\}) + 0P(\{\text{H wins from 8-7}\})$, so we need to compute $P(\{\text{T wins from 8-7}\})$; but $P(\{\text{T wins from 8-7}\}) = 1 - P(\{\text{H wins from 8-7}\})$. Now it is slightly easier to compute $P(\{\text{T wins from 8-7}\})$, because T can only win in two ways: 8-10 or 9-10. These are independent. For T to win 8-10, the next three flips must come up T, so that event has probability 1/8. For T to win 9-10, the next four flips must have one H in them, but the last flip may not be H (or else H wins); so the next four flips could be HTTT, THTT, or TTHT. The probability of this is 3/16. This means the total probability that T wins is 5/16. So T should get 16.625 and H should get the rest (although they might have to flip for the odd half cent).

6.3.4 Making a Decision with Decision Trees and Expectations

Imagine we have to choose an action. Once we have chosen, a sequence of random events occurs, and we get a reward with some probability. Which action should we choose? A good answer is to choose the action with the best expected outcome. In fact, choosing any other action is unwise, because if we encounter this situation repeatedly and make a choice that is even only slightly worse than the best, we could lose heavily. This is a very common recipe, and it can be applied to many situations. Usually, but not always, the reward is in money, and we will compute with money rewards for the first few examples.

For such problems, it can be useful to draw a **decision tree**. A decision tree is a drawing of possible outcomes of decisions, which makes costs, benefits and random elements explicit. Each node of the tree represents a test of an attribute (which could be either a decision, or a random variable), and each edge represents a possible outcome of a test. The final outcomes are leaves. Usually, decision nodes are drawn as squares, chance elements as circles, and leaves as triangles.

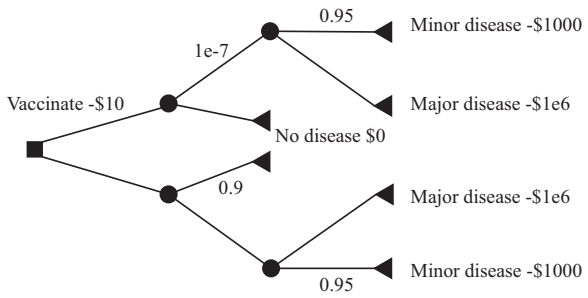

FIGURE 6.1: *A decision tree for the vaccination problem. The only decision is whether to vaccinate or not (the box at the root of the tree). I have only labelled edges where this is essential, so I did not annotate the "no vaccination" edge with zero cost. Once you decide whether to vaccinate or not, there is a random node (whether you get the disease or not), and, if you get it, another (minor or major).*

Worked example 6.16 *Vaccination*

It costs 10 to be vaccinated against a common disease. If you have the vaccination, the probability you will get the disease is $1e-7$. If you do not, the probability is 0.1. The disease is unpleasant; with probability 0.95, you will experience effects that cost you 1000 (eg several days in bed), but with probability 0.05, you will experience effects that cost you $1e6$. Should you be vaccinated?

Solution: Figure 6.1 shows a decision tree for this problem. I have annotated some edges with the choices represented, and some edges with probabilities; the sum of probabilities over all rightward (downgoing) edges leaving a random node is 1. It is straightforward to compute expectations. The expected cost of the disease is $0.95 \times 1000 + 0.05 \times 1e6 = 50,950$. If you are vaccinated, your expected income will be $-(10 + 1e-7 \times 50,950) = -10.01$ (rounding to the nearest cent). If you are not, your expected income is $-5,095$. You should be vaccinated.

Sometimes there is more than one decision. We can still do simple examples, though drawing a decision tree is now quite important, because it allows us to keep track of cases and avoid missing anything. For example, assume I wish to buy a cupboard. Two nearby towns have used furniture shops (usually called antique shops these days). One is further away than the other. If I go to town A, I will have time to look in two (of three) shops; if I go to town B, I will have time to look in one (of two) shops. I could lay out this sequence of decisions (which town to go to; which shop to visit when I get there) as Figure 6.2.

You should notice that this figure is missing a lot of information. What is the probability that I will find what I'm looking for in the shops? What is the value of

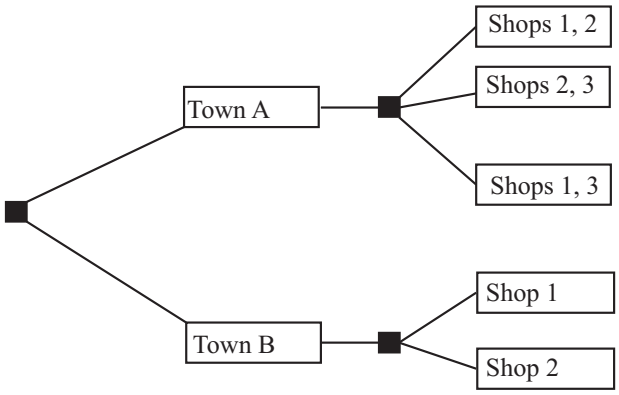

FIGURE 6.2: *The decision tree for the example of visiting furniture shops. Town A is nearer than town B, so if I go there I can choose to visit two of the three shops there; if I go to town B, I can visit only one of the two shops there. To decide what to do, I could fill in the probabilities and values of outcomes, compute the expected value of each pair of decisions, and choose the best. This could be tricky to do (where do I get the probabilities from?) but offers a rational and principled way to make the decision.*

finding it? What is the cost of going to each town? and so on. This information is not always easy to obtain. In fact, I might simply need to give my best subjective guess of these numbers. Furthermore, particularly if there are several decisions, computing the expected value of each possible sequence could get difficult. There are some kinds of model where one can compute expected values easily, but a good viable hypothesis about why people don't make optimal decisions is that optimal decisions are actually too hard to compute.

6.3.5 Utility

Sometimes it is hard to work with money. For example, in the case of a serious disease, choosing treatments often boils down to expected survival times, rather than money.

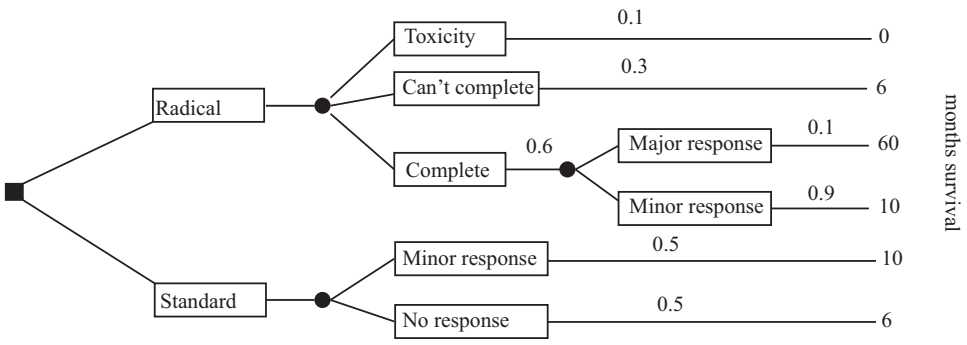

FIGURE 6.3: *A decision tree for example 17. Notations vary a bit, and here I have put boxes around the labels for the edges.*

Worked example 6.17 *Radical treatment*

(This example largely after Vickers, p97). Imagine you have a nasty disease. There are two kinds of treatment: standard, and radical. Radical treatment might kill you (with probability 0.1); might be so damaging that doctors stop (with probability 0.3); but otherwise you will complete the treatment. If you do complete radical treatment, there could be a major response (probability 0.1) or a minor response. If you follow standard treatment, there could be a major response (probability 0.5) or a minor response, but the outcomes are less good. All this is best summarized in a decision tree (Figure 6.3). What gives the longest expected survival time?

Solution: In this case, expected survival time with radical treatment is $(0.1 \times 0 + 0.3 \times 6 + 0.6 \times (0.1 \times 60 + 0.9 \times 10)) = 10.8$ months; expected survival time without radical treatment is $0.5 \times 10 + 0.5 \times 6 = 8$ months.

Working with money values is not always a good idea. For example, many people play state lotteries. The expected value of a 1 bet on a state lottery is well below 1 — why do people play? It's easy to assume that all players just can't do sums, but many players are well aware that the expected value of a bet is below the cost. It seems to be the case that people value money in a way that doesn't depend linearly on the amount of money. So, for example, people may value a million dollars rather more than a million times the value they place on one dollar. If this is true, we need some other way to keep track of value; this is sometimes called **utility**. It turns out to be quite hard to know how people value things, and there is quite good evidence that (a) human utility is complicated and (b) it is difficult to explain human decision making in terms of expected utility.

Worked example 6.18 *Human utility is not expected payoff*

Here are four games:

- **Game 1:** The player is given 1. A biased coin is flipped, and the money is taken back with probability p; otherwise, the player keeps it.
- **Game 2:** The player stakes 1, and a fair coin is flipped; if the coin comes up heads, the player gets r and the stake back, but otherwise loses the original stake.
- **Game 3:** The player bets nothing; a biased coin is flipped, and if it comes up heads (probability q), the player gets $1e6$.
- **Game 4:** The player stakes 1000; a fair coin is flipped, and if it comes up heads, the player gets s and the stake back, but otherwise loses the original stake.

In particular, what happens if $r = 3 - 2p$ and $q = (1 - p)/1e6$ and $s = 2 - 2p + 1000$?

Solution: Game 1 has expected value $(1-p)1$. Game 2 has expected value $(1/2)(r-1)$. Game 3 has expected value $q1e6$. Game 4 has expected value $(1/2)s - 500$.
In the case given, each game has the same expected value. Nonetheless, people usually have decided preferences for which game they would play. Generally, 4 is unattractive (seems expensive to play); 3 seems like free money, and so a good thing; 2 might be OK but is often seen as uninteresting; and 1 is unattractive. This should suggest to you that people's reasoning about money and utility is not what simple expectations can predict.

6.4 WHAT YOU SHOULD REMEMBER

You should be able to:

- Interpret notation for joint and conditional probability for random variables; in particular, understand notation such as: $P(\{X\})$, $P(\{X = x\})$, $p(x)$, $p(x, y)$, $p(x|y)$
- Interpret a probability density function $p(x)$ as $P(\{X \in [x, x + dx]\})$.
- Interpret the expected value of a discrete random variable.
- Interpret the expected value of a continuous random variable.
- Compute expected values of random variables for straightforward cases.

- Write down expressions for mean, variance and covariance for random variables.
- Write out a decision tree.

You should remember:

- The definition of a random variable.
- The definition of an expected value.
- The definitions of mean, variance and covariance.
- The definition of an indicator function.
- Bayes rule.
- The definition of marginalization.
- The Markov inequality.
- The Chebyshev Inequality.
- The weak law of large numbers.

PROBLEMS

Joint and Conditional Probability for Random Variables

6.1. Define a random variable X by the following procedure. Draw a card from a standard deck of playing cards. If the card is knave, queen, or king, then $X = 11$. If the card is an ace, then $X = 1$; otherwise, X is the number of the card (i.e. two through ten). Now define a second random variable Y by the following procedure. When you evaluate X, you look at the color of the card. If the card is red, then $Y = X - 1$; otherwise, $Y = X + 1$.
(a) What is $P(\{X \leq 2\})$?
(b) What is $P(\{X \geq 10\})$?
(c) What is $P(\{X \geq Y\})$?
(d) What is the probability distribution of $Y - X$?
(e) What is $P(\{Y \geq 12\}$?

6.2. Define a random variable by the following procedure. Flip a fair coin. If it comes up heads, the value is 1. If it comes up tails, roll a die: if the outcome is 2 or 3, the value of the random variable is 2. Otherwise, the value is 3.
(a) What is the probability distribution of this random variable?
(b) What is the cumulative distribution of this random variable?

6.3. Define three random variables, X, Y and Z by the following procedure. Roll a six-sided die and a four-sided die. Now flip a coin. If the coin comes up heads, then X takes the value of the six-sided die and Y takes the value of the four-sided die. Otherwise, X takes the value of the four-sided die and Y takes the value of the six-sided die. Z always takes the value of the sum of the dice.
(a) What is $P(X)$, the probability distribution of this random variable?
(b) What is $P(X, Y)$, the joint probability distribution of these two random variables?
(c) Are X and Y independent?

(d) Are X and Z independent?

6.4. Define two random variables X and Y by the following procedure. Flip a fair coin; if it comes up heads, then $X = 1$, otherwise $X = -1$. Now roll a six-sided die, and call the value U. We define $Y = U + X$.
(a) What is $P(Y|X = 1)$?
(b) What is $P(X|Y = 0)$?
(c) What is $P(X|Y = 7)$?
(d) What is $P(X|Y = 3)$?
(e) Are X and Y independent?

Expected Values

6.5. A simple coin game is as follows: we have a box, which starts empty. P1 flips a fair coin. If it comes up heads, P2 gets the contents of the box, and the game ends. If it comes up tails, P1 puts a dollar in the box and they flip again; this repeats until it comes up heads
(a) With what probability will P2 win exactly 10 units?
(b) What is the expected value of the game?
(c) How much should P2 pay to play, to make the game fair?

6.6. A simple card game is as follows. P1 pays a stake of 1 to play. P1 and P2 then each draw a card. If both cards are the same color, P2 keeps the stake and the game ends. If they are different colors, P2 pays P1 the stake and 1 extra (a total of 2).
(a) What is the expected value of the game to P1?
(b) P2 modifies the game, as follows. If both cards are court cards (that is, knave, queen, king), then P2 keeps the stake and the game ends; otherwise, the game works as before. Now what is the expected value of the game to P1?

6.7. An airline company runs a flight that has six seats. Each passenger who buys a ticket has a probability p of turning up for the flight. These events are independent.
(a) The airline sells six tickets. What is the expected number of passengers, if $p = 0.9$?
(b) How many tickets should the airline sell to ensure that the expected number of passengers is greater than six, if $p = 0.7$? **Hint:** The easiest way to do this is to write a quick program that computes the expected value of passengers that turn up for each the number of tickets sold, then search the number of tickets sold.

6.8. An airline company runs a flight that has 10 seats. Each passenger who buys a ticket has a probability p of turning up for the flight. The gender of the passengers is not known until they turn up for a flight, and women buy tickets with the same frequency that men do. The pilot is eccentric, and will not fly unless at least two women turn up.
(a) How many tickets should the airline sell to ensure that the expected number of passengers that turn up is greater than 10?
(b) The airline sells 10 tickets. What is the expected number of passengers on the aircraft, given that it flies? (i.e. that at least two women turn up). Estimate this value with a simulation.

Mean, Variance and Covariance

6.9. Show that $\mathsf{var}[kX] = k^2\mathsf{var}[X]$.

6.10. Show that if X and Y are independent random variables, then $\mathsf{var}[X+Y] = \mathsf{var}[X] + \mathsf{var}[Y]$. You will find it helpful to remember that, for X and Y independent, $\mathbb{E}[XY] = \mathbb{E}[X]\mathbb{E}[Y]$.

Using Inequalities

6.11. The random variable X takes the values -2, -1, 0, 1, 2, but has an unknown probability distribution. You know that $\mathbb{E}[\|X\|] = 0.2$. Use Markov's inequality to give a *lower* bound on $P(\{X = 0\})$. *Hint:* Notice that $P(\{X = 0\}) = 1 - P(\{\|X\| = 1\}) - P(\{\|X\|\} = 2)$.

6.12. The random variable X takes the values 1, 2, 3, 4, 5, but has unknown probability distribution. You know that $\mathbb{E}[X] = 2$ and $\mathsf{var}(\{X\}) = 0.01$. Use Chebychev's inequality to give a *lower* bound on $P(\{X = 2\})$.

Using Expectations

6.13. Imagine we have a game with two players, who are playing for a stake. There are no draws, the winner gets the whole stake, and the loser gets nothing. The game must end early. We decide to give each player the expected value of the game for that player, from that state. Show that the expected values add up to the value of the stake (i.e. there won't be too little or too much money in the stake.

General Exercises

CHAPTER 7

Useful Probability Distributions

In most practical problems, we observe some data, then try to infer properties of the process or processes that produced the data. For example, we might ask what new data would look like, or whether two datasets come from the same or different sources. When we do this, we are usually trying to determine some properties of the probability distribution of a random variable. The data represent values that the random variable has taken. In many cases, the probability distribution involved can be modelled using one of a small set of standard probability distributions. In this chapter, I collect facts about the distributions we will use in following chapters.

7.1 DISCRETE DISTRIBUTIONS

7.1.1 The Discrete Uniform Distribution

If every value of a discrete random variable has the same probability, then the probability distribution is the discrete uniform distribution. We have seen this distribution before, numerous times. For example, I define a random variable by the number that shows face-up on the throw of a die. This has a uniform distribution. As another example, write the numbers 1-52 on the face of each card of a standard deck of playing cards. The number on the face of the first card drawn from a well-shuffled deck is a random variable with a uniform distribution.

One can construct expressions for the mean and variance of a discrete uniform distribution, but they're not usually much use (too many terms, not often used). Keep in mind that if two random variables have a uniform distribution, their sum and difference will not (recall example 3).

7.1.2 The Geometric Distribution

We have a biased coin. The probability it will land heads up, $P(\{H\})$ is given by p. We flip this coin until the first head appears. The number of flips required is a discrete random variable which takes integer values greater than or equal to one, which we shall call X. To get n flips, we must have $n-1$ tails followed by 1 head. This event has probability $(1-p)^{(n-1)}p$. We can now write out the probability distribution that n flips are required.

Definition: 7.1 *The Geometric Distribution*

We have an experiment with a binary outcome (i.e. heads or tails; 0 or 1; and so on), with $P(H) = p$ and $P(T) = 1 - p$. We repeat this experiment until the first head occurs. The probability distribution for n, the number of repetitions, is the geometric distribution. It has the form

$$P(\{X = n\}) = (1 - p)^{(n-1)}p.$$

for $0 \leq p \leq 1$ and $n \geq 1$; for other n it is zero. p is called the **parameter** of the distribution.

Notice that the geometric distribution is non-negative everywhere. It is straightforward to show that it sums to one, and so is a probability distribution (exercises).

Useful Facts: 7.1 *The geometric distribution*

1. The mean of the geometric distribution is $\frac{1}{p}$.
2. The variance of the geometric distribution is $\frac{1-p}{p^2}$.

The proof of these facts requires some work with series, and is relegated to the exercises.

7.1.3 Bernoulli Random Variables

A Bernoulli random variable models a biased coin with probability p of coming up heads in any one flip.

Definition: 7.2 *Bernoulli Random Variable*

A Bernoulli random variable takes the value 1 with probability p and 0 with probability $1 - p$. This is a model for a coin toss, among other things

Useful Facts: 7.2 *Bernoulli Random Variables*

1. A Bernoulli random variable has mean p.
2. A Bernoulli random variable has variance $p(1-p)$.

Proofs are easy, and in the exercises.

7.1.4 The Binomial Probability Distribution

Assume we have a biased coin with probability p of coming up heads in any one flip. The binomial probability distribution gives the probability that it comes up heads h times in N flips.

Worked example 32 yields one way of deriving this distribution. In that example, I showed that there are

$$N!/(h!(N-h)!)$$

outcomes of N coin flips that have h heads. These outcomes are disjoint, and each has probability $p^h(1-p)^{(N-h)}$. As a result, we must have the probability distribution below.

Definition: 7.3 *The Binomial distribution*

In N independent repetitions of an experiment with a binary outcome (ie heads or tails; 0 or 1; and so on) with $P(H) = p$ and $P(T) = 1-p$, the probability of observing a total of h H's and $N-h$ T's is

$$P_b(h; N, p) = \left(\begin{array}{c} N \\ h \end{array} \right) p^h (1-p)^{(N-h)}$$

as long as $0 \leq h \leq N$; in any other case, the probability is zero.

Useful Fact: 7.3 *The binomial distribution*

Write $P_b(i; N, p)$ for the binomial distribution that one observes i H's in N trials.

$$\sum_{i=0}^{N} P_b(i; N, p) = (p + (1-p))^N = (1)^N = 1$$

by pattern matching to the binomial theorem. As a result,

$$\sum_{i=0}^{N} P_b(i; N, p) = 1$$

The binomial distribution satisfies a recurrence relation. We must have that

$$P_b(h; N, p) = pP_b(h-1; N-1, p) + (1-p)P_b(h; N-1, p).$$

This is because can get h heads in N flips either by having $h-1$ heads in $N-1$ flips, then flipping another, or by having h heads in N flips then flipping a tail. You can verify by induction that the binomial distribution satisfies this recurrence relation.

Useful Facts: 7.4 *The binomial distribution*

1. The mean of $P_b(i; N, p)$ is Np.
2. The variance of $P_b(i; N, p)$ is $Np(1-p)$

The proofs are informative, and so are not banished to the exercises.

Proof: 7.1 *The binomial distribution*

Notice that the number of heads in N coin tosses is can be obtained by adding the number of heads in each toss. Write Y_i for the Bernoulli random variable representing the i'th toss. If the coin comes up heads, $Y_i = 1$, otherwise $Y_i = 0$. The Y_i are independent. Now

$$\begin{aligned}
\mathbb{E}[X] &= \mathbb{E}\left[\sum_{j=1}^{N} Y_i\right] \\
&= \sum_{j=1}^{N} \mathbb{E}[Y_i] \\
&= N\mathbb{E}[Y_1] \qquad \text{because the } Y_i \text{ are independent} \\
&= Np.
\end{aligned}$$

The variance is easy, too. Each coin toss is independent, so the variance of the sum of coin tosses is the sum of the variances. This gives

$$\begin{aligned}
\mathsf{var}[X] &= \mathsf{var}\left[\sum_{j=1}^{N} Y_i\right] \\
&= N\mathsf{var}[Y_1] \\
&= Np(1-p)
\end{aligned}$$

7.1.5 Multinomial probabilities

The binomial distribution describes what happens when a coin is flipped multiple times. But we could toss a die multiple times too. Assume this die has k sides, and we toss it N times. The distribution of outcomes is known as the **multinomial distribution**.

We can guess the form of the multinomial distribution in rather a straightforward way. The die has k sides. We toss the die N times. This gives us a sequence of N numbers. Each toss of the die is independent. Assume that side 1 appears n_1 times, side 2 appears n_2 times, ... side k appears n_k times. Any single sequence with this property will appear with probability $p_1^{n_1} p_2^{n_2} ... p_k^{n_k}$, because the tosses are independent. However, there are

$$\frac{N!}{n_1! n_2! ... n_k!}$$

such sequences. Using this reasoning, we arrive at the distribution below

Definition: 7.4 *Multinomial Distribution*

I perform N independent repetitions of an experiment with k possible outcomes. The i'th such outcome has probability p_i. I see outcome 1 n_1 times, outcome 2 n_2 times, etc. Notice that $n_1+n_2+n_3+\ldots+n_k = N$. The probability of observing this set of outcomes is

$$P_m(n_1,\ldots,n_k;N,p_1,\ldots,p_k) = \frac{N!}{n_1!n_2!\ldots n_k!}p_1^{n_1}p_2^{n_2}\ldots p_k^{n_k}.$$

Worked example 7.1 *Dice*

I throw five fair dice. What is the probability of getting two 2's and three 3's?

Solution: $\frac{5!}{2!3!}(\frac{1}{6})^2(\frac{1}{6})^3$

7.1.6 The Poisson Distribution

Assume we are interested in counts that occur in an interval of time (e.g. within a particular hour). Because they are counts, they are non-negative and integer valued. We know these counts have two important properties. First, they occur with some fixed average rate. Second, an observation occurs independent of the interval since the last observation. Then the Poisson distribution is an appropriate model.

There are numerous such cases. For example, the marketing phone calls you receive during the day time are likely to be well modelled by a Poisson distribution. They come at some average rate — perhaps 5 a day as I write, during the last phases of an election year — and the probability of getting one clearly doesn't depend on the time since the last one arrived. Classic examples include the number of Prussian soldiers killed by horse-kicks each year; the number of calls arriving at a call center each minute; the number of insurance claims occurring in a given time interval (outside of a special event like a hurricane, etc.).

Definition: 7.5 *The Poisson Distribution*

A non-negative, integer valued random variable X has a Poisson distribution when its probability distribution takes the form

$$P(\{X = k\}) = \frac{\lambda^k e^{-\lambda}}{k!},$$

where $\lambda > 0$ is a parameter often known as the **intensity** of the distribution.

Notice that the Poisson distribution is a probability distribution, because it is non-negative and because

$$\sum_{i=0}^{\infty} \frac{\lambda^i}{i!} = e^{\lambda}$$

so that

$$\sum_{k=0}^{\infty} \frac{\lambda^k e^{-\lambda}}{k!} = 1$$

Useful Facts: 7.5 *The Poisson Distribution*

1. The mean of a Poisson distribution with intensity λ is λ.
2. The variance of a Poisson distribution with intensity λ is λ (no, that's not an accidentally repeated line or typo).

The proof of these facts requires some work with series, and is relegated to the exercises.

I described the Poisson distribution as a natural model for counts of randomly distributed points along a time axis. But it doesn't really matter that this is a time axis — it could be a space axis instead. For example, you could take a length of road, divide it into even intervals, then count the number of road-killed animals is in each interval. If the location of each animal is independent of the location of any other animal, then you could expect a Poisson model to apply to the count data. Assume that the Poisson model that best describes the data has parameter λ. One property of such models is that if you doubled the length of the intervals, then the resulting dataset would be described by a Poisson model with parameter 2λ; similarly, if you halved the length of the intervals, the best model would have parameter $\lambda/2$. This corresponds to our intuition about such data; roughly, the number of road-killed animals in two miles of road should be twice the number in one mile of road. This property means that no pieces of the road are "special" — each behaves the same as the other.

We can build a really useful model of spatial randomness by observing this fact and generalizing very slightly. A **Poisson point process** with intensity λ is a set of random points with the property that the number of points in an interval of length s is a Poisson random variable with parameter λs. Notice how this captures our intuition that if points are "very randomly" distributed, there should be twice as many of them in an interval that is twice as long.

This model is easily, and very usefully, extended to points on the plane, on surfaces, and in 3D. In each case, the process is defined on a domain D (which has to meet some very minor conditions that are of no interest to us). The number of points in any subset s of D is a Poisson random variable, with intensity $\lambda m(s)$, where $m(s)$ is the area (resp. volume) of s. These models are useful, because they capture the property that (a) the points are random and (b) the probability you find a point doesn't depend on where you are. You could reasonably believe models like this apply to, say, dead flies on windscreens; the places where you find acorns at the foot of an oak tree; the distribution of cowpats in a field; the distribution of cherries in a fruitcake; and so on.

7.2 CONTINUOUS DISTRIBUTIONS

7.2.1 The Continuous Uniform Distribution

Some continuous random variables have a natural upper bound and a natural lower bound but otherwise we know nothing about them. For example, imagine we are given a coin of unknown properties by someone who is known to be a skillful maker of unfair coins. The manufacturer makes no representations as to the behavior of the coin. The probability that this coin will come up heads is a random variable, about which we know nothing except that it has a lower bound of zero and an upper bound of one.

If we know nothing about a random variable apart from the fact that it has a lower and an upper bound, then a **uniform distribution** is a natural model. Write l for the lower bound and u for the upper bound. The probability density function for the uniform distribution is

$$p(x) = \begin{cases} 0 & x < l \\ 1/(u-l) & l \leq x \leq u \\ 0 & x > u \end{cases}$$

A continuous random variable whose probability distribution is the uniform distribution is often called a **uniform random variable**.

7.2.2 The Beta Distribution

It's hard to explain now why the Beta (or β) distribution is useful, but it will come in useful later (section 1). The Beta distribution is a probability distribution for a continuous random variable x in the range $0 \leq x \leq 1$. There are two parameters, $\alpha > 0$ and $\beta > 0$. Recall the definition of the Γ function from section 1.1. We have that

$$P_\beta(x|\alpha, \beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{(\alpha-1)}(1-x)^{(\beta-1)}.$$

From this expression, you can see that:

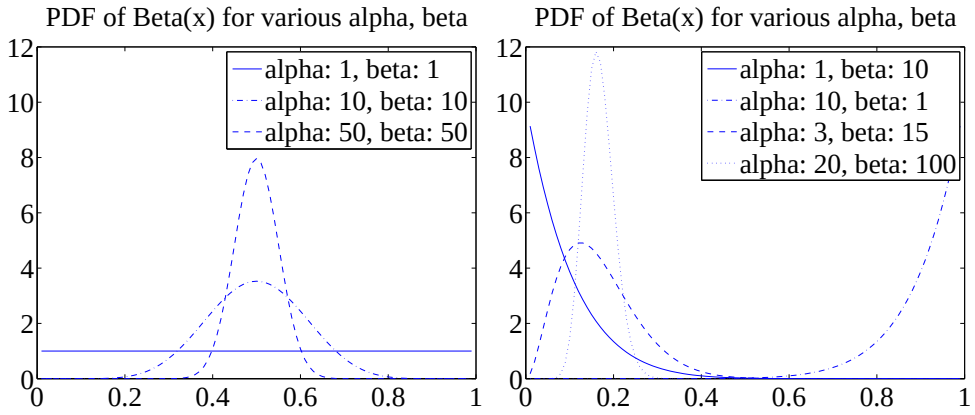

FIGURE 7.1: *Probability density functions for the Beta distribution with a variety of different choices of α and β.*

- $P_\beta(x|1,1)$ is a uniform distribution on the unit interval.
- $P_\beta(x|\alpha,\beta)$ has a single maximum at $x = (\alpha-1)/(\alpha+\beta-2)$ for $\alpha > 1$, $\beta > 1$) (differentiate and set to zero).
- Generally, as α and β get larger, this peak gets narrower.
- For $\alpha = 1$, $\beta > 1$ the largest value of $P_\beta(x|\alpha,\beta)$ is at $x = 0$.
- For $\alpha > 1$, $\beta = 1$ the largest value of $P_\beta(x|\alpha,\beta)$ is at $x = 1$.

Figure 7.1 shows plots of the probability density function of the Beta distribution for a variety of different values of α and β.

Useful Facts: 7.6 *The Beta Distribution*

For a Beta distribution with parameters α, β

1. The mean is $\frac{\alpha}{\alpha+\beta}$.
2. The variance is $\frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$.

7.2.3 The Gamma Distribution

The Gamma (or γ) distribution will also come in useful later on (section 1). The Gamma distribution is a probability distribution for a non-negative continuous random variable $x \geq 0$. There are two parameters, $\alpha > 0$ and $\beta > 0$. The probability density function is

$$P_\gamma(x|\alpha,\beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{(\alpha-1)} e^{-\beta x}.$$

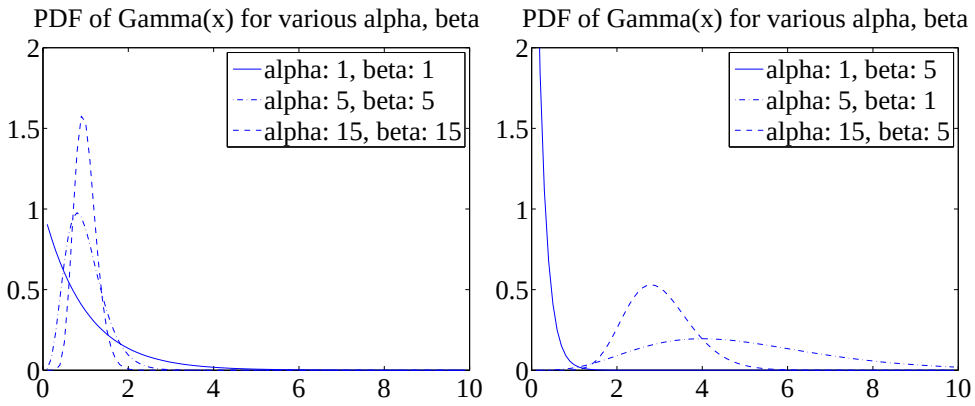

FIGURE 7.2: *Probability density functions for the Gamma distribution with a variety of different choices of α and β.*

Figure 7.2 shows plots of the probability density function of the Gamma distribution for a variety of different values of α and β.

Useful Facts: 7.7 *The Gamma Distribution*

For a Gamma distribution with parameters α, β

1. The mean is $\frac{\alpha}{\beta}$.
2. The variance is $\frac{\alpha}{\beta^2}$.

7.2.4 The Exponential Distribution

Assume we have an infinite interval of time or space, with points distributed on it. Assume these points form a Poisson point process, as above. For example, we might consider the times at which email arrives; or the times at which phone calls arrive at a large telephone exchange; or the locations of roadkill on a road. The distance (or span of time) between two consecutive points is a random variable X. This random variable takes an **exponential distribution**. There is a single parameter, λ. We have that

$$P_{\exp}(x|\lambda) = \begin{cases} \lambda \exp^{-\lambda x} & \text{for } x \geq 0 \\ 0 & \text{otherwise} \end{cases}.$$

This distribution is often useful in modelling the failure of objects. We assume that failures form a Poisson process in time; then the time to the next failure is exponentially distributed.

Useful Facts: 7.8 *The Exponential Distribution*

For an exponential distribution with parameter λ

1. The mean is
$$\frac{1}{\lambda}.$$
2. The variance is
$$\frac{1}{\lambda^2}.$$

Notice the relationship between this parameter and the parameter of the Poisson distribution. If (say) the phone calls are distributed with Poisson distribution with intensity λ (per hour), then your expected number of calls per hour is λ. The time between calls will be exponentially distributed with parameter λ, and the expected time to the next call is $1/\lambda$ (in hours).

7.3 THE NORMAL DISTRIBUTION

7.3.1 The Standard Normal Distribution

Definition: 7.6 *The Standard Normal Distribution*

The probability density function

$$p(x) = \left(\frac{1}{\sqrt{2\pi}}\right) \exp\left(\frac{-x^2}{2}\right).$$

is known as the **standard normal distribution**

The first step is to plot this probability density function (Figure 7.3). You should notice it is quite familiar from work on histograms, etc. in Chapter 1. It has the shape of the histogram of standard normal data, or at least the shape that the histogram of standard normal data aspires to.

Useful Facts: 7.9 *The standard normal distribution*

1. The mean of the standard normal distribution is 0.
2. The variance of the standard normal distribution is 1.

These results are easily established by looking up (or doing!) the relevant integrals; they are relegated to the exercises.

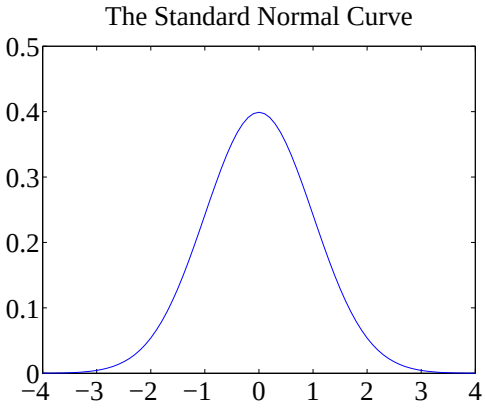

FIGURE 7.3: *A plot of the probability density function of the standard normal distribution. Notice how probability is concentrated around zero, and how there is relatively little probability density for numbers with large absolute values.*

A continuous random variable is a **standard normal random variable** if its probability density function is a standard normal distribution.

7.3.2 The Normal Distribution

Any probability density function that is a standard normal distribution *in standard coordinates* is a **normal distribution**. Now write μ for the mean of a random variable and σ for its standard deviation; we are saying that, if

$$\frac{x-\mu}{\sigma}$$

has a standard normal distribution, then $p(x)$ is a normal distribution. We can work out the form of the probability density function of a general normal distribution in two steps: first, we notice that for any normal distribution, we must have

$$p(x) \propto \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right].$$

But, for this to be a probability density function, we must have $\int_{-\infty}^{\infty} p(x)dx = 1$. This yields the constant of proportionality, and we get

Definition: 7.7 *The Normal Distribution*

The probability density function

$$p(x) = \left(\frac{1}{\sqrt{2\pi}\sigma}\right)\exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right).$$

is a normal distribution.

Useful Facts: 7.10 *The normal distribution*

The probability density function

$$p(x) = \left(\frac{1}{\sqrt{2\pi}\sigma}\right) \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right).$$

has

1. mean μ
2. and variance σ.

These results are easily established by looking up (or doing!) the relevant integrals; they are relegated to the exercises.

A continuous random variable is a **normal random variable** if its probability density function is a **normal distribution**. Notice that it is quite usual to call normal distributions **gaussian distributions**.

7.3.3 Properties of the Normal Distribution

Normal distributions are important, because one often runs into data that is well described by a normal distribution. It turns out that anything that behaves like a binomial distribution with a lot of trials — for example, the number of heads in many coin tosses; as another example, the percentage of times you get the outcome of interest in a simulation in many runs — should produce a normal distribution (Section 7.4). For this reason, pretty much any experiment where you perform a simulation, then count to estimate a probability or an expectation, should give you an answer that has a normal distribution.

It is a remarkable and deep fact, known as the **central limit theorem**, that adding many independent random variables produces a normal distribution pretty much *whatever* the distributions of those random variables. I've not shown this in detail because it's a nuisance to prove. However, if you add together many random variables, each of pretty much any distribution, then the answer has a distribution close to the normal distribution. It turns out that many of the processes we observe add up subsidiary random variables. This means that you will see normal distributions very often in practice.

A normal random variable tends to take values that are quite close to the mean, measured in standard deviation units. We can demonstrate this important fact by computing the probability that a standard normal random variable lies between u and v. We form

$$\int_u^v \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{u^2}{2}\right) du.$$

It turns out that this integral can be evaluated relatively easily using a special

function. The **error function** is defined by

$$\mathsf{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x \exp\left(-t^2\right) dt$$

so that

$$\frac{1}{2}\mathsf{erf}\left((\frac{x}{\sqrt{2}})\right) = \int_0^x \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{u^2}{2}\right) du.$$

Notice that $\mathsf{erf}(x)$ is an odd function (i.e. $\mathsf{erf}(-x) = \mathsf{erf}(x)$). From this (and tables for the error function, or Matlab) we get that, for a standard normal random variable

$$\frac{1}{\sqrt{2\pi}} \int_{-1}^{1} \exp\left(-\frac{x^2}{2}\right) dx \approx 0.68$$

and

$$\frac{1}{\sqrt{2\pi}} \int_{-2}^{2} \exp\left(-\frac{x^2}{2}\right) dx \approx 0.95$$

and

$$\frac{1}{\sqrt{2\pi}} \int_{-2}^{2} \exp\left(-\frac{x^2}{2}\right) dx \approx 0.99.$$

These are very strong statements. They measure how often a standard normal random variable has values that are in the range $-1, 1$, $-2, 2$, and $-3, 3$ respectively. But these measurements apply to normal random variables if we recognize that they now measure how often the normal random variable is some number of standard deviations away from the mean. In particular, it is worth remembering that:

Useful Facts: 7.11 *Normal Random Variables*

- About 68% of the time, a normal random variable takes a value within one standard deviation of the mean.
- About 95% of the time, a normal random variable takes a value within one standard deviation of the mean.
- About 99% of the time, a normal random variable takes a value within one standard deviation of the mean.

7.4 APPROXIMATING BINOMIALS WITH LARGE N

The Binomial distribution appears to be a straightforward thing. We assume we flip a coin N times, where N is a very large number. The coin has probability p of coming up heads, and so probability $q = 1 - p$ of coming up tails. The number of heads h follows the binomial distribution, so

$$P(h) = \frac{N!}{h!(N-h)!} p^h q^{(N-h)}$$

The mean of this distribution is Np, the variance is Npq, and the standard deviation is $\sqrt{Npq}$.

Evaluating this probability distribution for large N is very difficult, because factorials grow fast. We will construct an approximation to the binomial distribution for large N that allows us to evaluate the probability that h lies in some range. This approximation will show that the probability that h is within one standard deviation of the mean is approximately 68%.

This is important, because it shows that our model of probability as frequency is consistent. Consider the probability that the number of heads you see lies within one standard deviation of the mean. The size of that interval is $2\sqrt{Npq}$. As N gets bigger, the size of that interval, *relative to the total number of flips*, gets smaller. If I flip a coin N times, in principle I could see a number of heads h that ranges from 0 to N. However, we will establish that about 68% of the time, h will lie in the interval within one standard deviation of the mean. The size of this interval, *relative to the total number of flips* is

$$2\frac{\sqrt{Npq}}{N} = 2\sqrt{\frac{pq}{N}}.$$

As a result, as $N \to \infty$,

$$\frac{h}{N} \to p$$

because h will tend to land in an interval around pN that gets narrower as N gets larger.

The main difficulty with Figure 7.4 (and with the argument above) is that the mean and standard deviation of the binomial distribution tends to infinity as the number of coin flips tends to infinity. This can confuse issues. For example, the plots of Figure 7.4 show narrowing probability distributions — but is this because the scale is compacted, or is there a real effect? It turns out there is a real effect, and a good way to see it is to consider the normalized number of heads.

7.4.1 Large N

Recall that to normalize a dataset, you subtract the mean and divide the result by the standard deviation. We can do the same for a random variable. We now consider

$$x = \frac{h - Np}{\sqrt{Npq}}.$$

The probability distribution of x can be obtained from the probability distribution for h, because $h = Np + x\sqrt{Npq}$, so

$$P(x) = \left(\frac{N!}{(Np + x\sqrt{Npq})!(Nq - x\sqrt{Npq})!} \right) p^{(Np + x\sqrt{Npq})} q^{(Nq - x\sqrt{Npq})}.$$

I have plotted this probability distribution for various values of N in Figure 7.5.

But it is hard to work with this distribution for very large N. The factorials become very difficult to evaluate. Second, it is a discrete distribution on N points, spaced $1/\sqrt{Npq}$ apart. As N becomes very large, the number of points that have

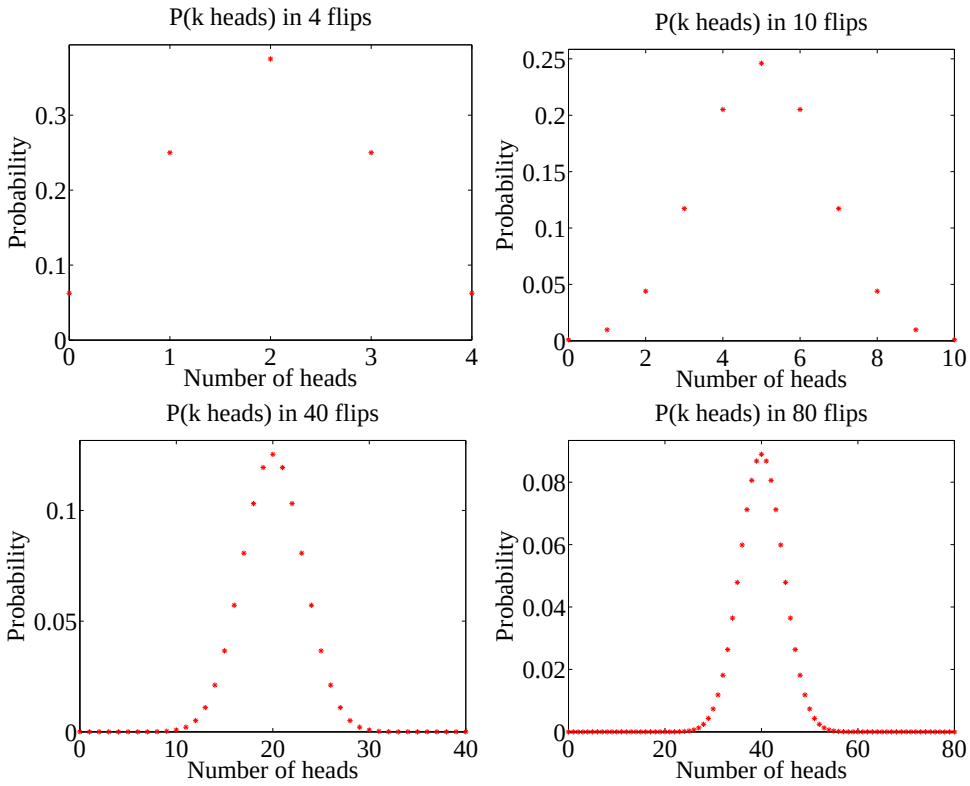

FIGURE 7.4: *Plots of the binomial distribution for $p = q = 0.5$ for different values of N. You should notice that the set of values of h (the number of heads) that have substantial probability is quite narrow compared to the range of possible values. This set gets narrower as the number of flips increases. This is because the mean is pN and the standard deviation is $\sqrt{Npq}$ — so the fraction of values that is within one standard deviation of the mean is $O(1/\sqrt{N})$.*

non-zero probability becomes very large, and x can be very large, or very small. For example, there is some probability, though there may be very little indeed, on the point where $h = N$, or, equivalently, $x = N(p + \sqrt{Npq})$. For sufficiently large N, we think of this probability distribution as a probability density function. We can do so, for example, by spreading the probability for x_i (the i'th value of x) evenly over the interval between x_i and x_{i+1}. We then have a probability density function that looks like a histogram, with bars that become narrower as N increases. But what is the limit?

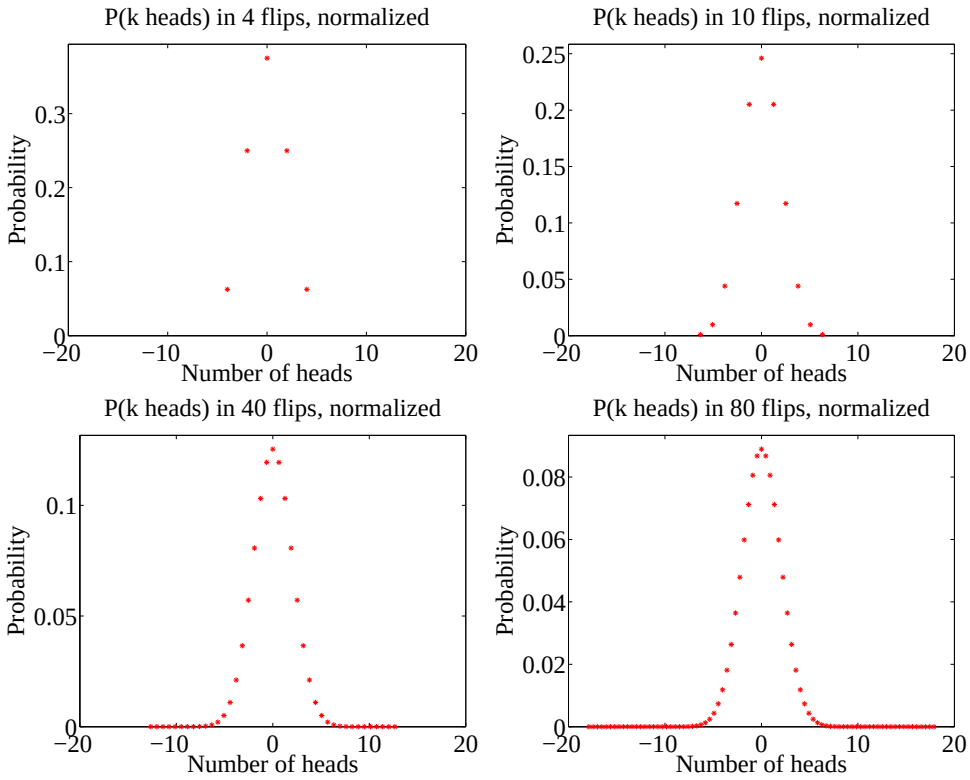

FIGURE 7.5: *Plots of the distribution for the normalized variable x, with $P(x)$ given in the text, obtained from the binomial distribution with $p = q = 0.5$ for different values of N. These distributions are normalized (mean 0, variance 1. They look increasingly like a standard normal distribution EXCEPT that the value at their mode gets smaller as N gets bigger (there are more possible outcomes). In the text, we will establish that the standard normal distribution is a limit, in a useful sense.*

7.4.2 Getting Normal

To proceed, we need Stirling's approximation, which says that, for large N,

$$N! \approx \sqrt{2\pi}\sqrt{N}\left(\frac{N}{e}\right)^N.$$

This yields

$$P(h) \approx \left(\frac{Np}{h}\right)^h \left(\frac{Nq}{N-h}\right)^{(N-h)} \sqrt{\frac{N}{2\pi h(N-h)}}$$

Recall we used the normalized variable

$$x = \frac{h - Np}{\sqrt{Npq}}.$$

We will encounter the term $\sqrt{Npq}$ often, and we use $\sigma = \sqrt{Npq}$ as a shorthand. We can compute h and $N - h$ from x by the equalities

$$h = Np + \sigma x \qquad\qquad N - h = Nq - \sigma x.$$

So the probability distribution written in this new variable x is

$$P(x) \approx \left(\frac{Np}{(Np+\sigma x)}\right)^{(Np+\sigma x)} \left(\frac{Nq}{(Nq-\sigma x)}\right)^{(Nq-\sigma x)} \sqrt{\frac{N}{2\pi(Np+\sigma x)(Nq-\sigma x)}}$$

There are three terms to deal with here. It is easiest to work with $\log P$. Now

$$\log(1+x) = x - \frac{1}{2}x^2 + O(x^3)$$

so we have

$$\begin{aligned} \log\left(\frac{Np}{(Np+\sigma x)}\right) &= -\log\left(1+\frac{\sigma x}{Np}\right) \\ &\approx -\frac{\sigma x}{Np} + (\frac{1}{2})(\frac{\sigma x}{Np})^2 \end{aligned}$$

and

$$\log\left(\frac{Nq}{(Nq-\sigma x)}\right) \approx \frac{\sigma x}{Nq} + (\frac{1}{2})(\frac{\sigma x}{Nq})^2.$$

From this, we have that

$$\begin{aligned} \log\left[\left(\frac{Np}{Np+\sigma x}\right)^{(Np+\sigma x)} \left(\frac{Nq}{Nq-\sigma x}\right)^{(Nq-\sigma x)}\right] &\approx [Np+\sigma x]\left[-\frac{\sigma x}{Np} + \left(\frac{1}{2}\right)\left(\frac{\sigma x}{Np}\right)^2\right] + \\ &\quad [Nq-\sigma x]\left[\frac{\sigma x}{Nq} + \left(\frac{1}{2}\right)\left(\frac{\sigma x}{Nq}\right)^2\right] \\ &= -\left(\frac{1}{2}\right)x^2 + O((\sigma x)^3) \end{aligned}$$

(recall $\sigma = \sqrt{Npq}$ if you're having trouble with the last step). Now we look at the square-root term. We have

$$\begin{aligned} \log\sqrt{\frac{N}{2\pi(Np+\sigma x)(Nq-\sigma x)}} &= -\frac{1}{2}\left(\log[Np+\sigma x] + \log[Nq-\sigma x] - \log N + \log 2\pi\right) \\ &= -\frac{1}{2}\begin{pmatrix} \log Np + O\left(\left(\frac{\sigma x}{Np}\right)\right) \\ +\log Nq - O\left(\left(\frac{\sigma x}{Nq}\right)\right) \\ -\log N + \log 2\pi \end{pmatrix} \end{aligned}$$

but, since N is very large compared to σx, we can ignore the $O(\left(\frac{\sigma x}{Np}\right))$ terms. Then this term is not a function of x. So we have

$$\log P(x) \approx \frac{-x^2}{2} + \text{constant}.$$

Now because N is very large, our probability distribution P limits to a probability density function p, with

$$p(x) \propto \exp\left(\frac{-x^2}{2}\right).$$

We can get the constant of proportionality from integrating, to

$$p(x) = \left(\frac{1}{\sqrt{2\pi}}\right)\exp\left(\frac{-x^2}{2}\right).$$

This constant of proportionality deals with the effect in figure 7.5, where the mode of the distribution gets smaller as N gets bigger. It does so because there are more points with non-zero probability to be accounted for. But we are interested in the limit where N tends to infinity. This must be a probability density function, so it must integrate to one.

Review this blizzard of terms. We started with a binomial distribution, but standardized the variables so that the mean was zero and the standard deviation was one. We then assumed there was a very large number of coin tosses, so large that that the distribution started to look like a continuous function. The function we get is the standard normal distribution.

7.4.3 So What?

I have proven an extremely useful fact, which I shall now put in a box.

Useful Fact: 7.12 *The Binomial Distribution for Large N*

Assume h follows the binomial distribution with parameters p and q. Write $x = \frac{h-Np}{\sqrt{Npq}}$. Then, for sufficiently large N, the probability distribution $P(x)$ can be approximated by the probability density function

$$\left(\frac{1}{\sqrt{2\pi}}\right)\exp\left(\frac{-x^2}{2}\right)$$

in the sense that

$$P(\{x \in [a,b]\}) \approx \int_a^b \left(\frac{1}{\sqrt{2\pi}}\right)\exp\left(\frac{-u^2}{2}\right)du$$

This justifies our model of probability as frequency. I interpreted an event having probability p to mean that, if I had a large number N of independent repetitions of the experiment, the number that produced the event would be close to Np, and would get closer as N got larger. We know that, for example, 68% of the time a standard normal random variable takes a value between 1 and -1. In this case, the standard normal random variable is

$$\frac{h-(Np)}{\sqrt{Npq}}$$

so that 68% of the time, h must take a value in the range $[Np-\sqrt{Npq}, Np+\sqrt{Npq}]$. Equivalently, the relative frequency h/N must take a value in the range

$$[p - \frac{pq}{\sqrt{N}}, p + \frac{pq}{\sqrt{N}}]$$

but as $N \to \infty$ this range gets smaller and smaller, and h/N limits to p. So our view of probability as a frequency is consistent.

To obtain h, we added N independent Bernoulli random variables. So you can interpret the box as saying that the sum of many independent Bernoulli random variables has a probability distribution that limits to the normal distribution as the number added together gets larger. Remember that I have stated, though not precisely, but not proved the deep and useful fact that the sum of pretty much any independent random variables has a distribution that gets closer to a normal distribution as the number added together gets larger.

7.5 WHAT YOU SHOULD REMEMBER

You should remember:

- The form of the uniform distribution.
- The form of the geometric distribution, and its mean and variance.
- The form of the binomial distribution, and its mean and variance.
- The form of the Poisson distribution, and its mean and variance.
- The form of the Normal distribution, and its mean and variance.
- The fact that a sum of a large number of IID binomial random variables is normally distributed, and the mean and variance of that distribution.
- The fact that a sum of a large number of IID random variables is normally distributed for most cases you will encounter.

PROBLEMS

Sums and Differences of Discrete Random Variables

7.1. Assume X and Y are discrete random variables which take integer values in the range $1 \ldots 100$ (inclusive). Write $S = X + Y$.

(a) Show that

$$P(S = k) = \sum_{u=1}^{u=100} P(\{\{X = k - u\} \cap \{Y = u\}\}).$$

(b) Show that

$$P(D = k) = \sum_{u=1}^{u=100} P(\{X = k + u\})P(\{Y = u\}).$$

(c) Now assume that both X and Y are uniform random variables. Show that S is not uniform by considering $P(S = 2)$, $P(S = 3)$, and $P(S = 100)$.

The Geometric Distribution

7.2. Recall that for $0 < r < 1$,

$$\sum_{i=0}^{\infty} r^i = \frac{1}{1-r}.$$

Now show that

$$\begin{aligned}\sum_{n=1}^{\infty} P(\{X = n\}) &= p\sum_{n=1}^{\infty}(1-p)^{(n-1)} \\ &= 1\end{aligned}$$

7.3. Write $S_\infty = \sum_{i=0}^{\infty} r^i$. Show that $(1-r)S_\infty = 1$, so that

$$S_\infty = \frac{1}{1-r}$$

7.4. Show that

$$\sum_{i=0}^{\infty} ir^i = (\sum_{i=1}^{\infty} r^i) + r(\sum_{i=1}^{\infty} r^i) + r^2(\sum_{i=1}^{\infty} r^i) + \ldots$$

(look carefully at the limits of the sums!) and so show that

$$\sum_{i=0}^{\infty} ir^i = \frac{r}{(1-r)^2}.$$

7.5. Write $S_\infty = \sum_{i=0}^{\infty} r^i$. Show that

$$\sum_{i=0}^{\infty} i^2 r^i = (\gamma - 1) + 3r(\gamma - 1) + 5r^2(\gamma - 1) + \ldots$$

and so that

$$\sum_{i=0}^{\infty} i^2 r^i = \frac{r(1+r)}{(1-r)^3}$$

7.6. Show that, for a geometric distribution with parameter p, the mean is

$$\sum_{i=1}^{\infty} i(1-p)^{(i-1)}p = \sum_{i=0}^{\infty}(i+1)(1-p)^i p.$$

Now by rearranging and using the previous results, show that the mean is

$$\sum_{i=1}^{\infty} i(1-p)^{(i-1)}p = \frac{1}{p}$$

7.7. Show that, for a geometric distribution with parameter p, the variance is $\frac{1-p}{p^2}$. To do this, note the variance is $\mathbb{E}\left[X^2\right] - \mathbb{E}[X]^2$. Now use the results of the previous exercises to show that

$$\mathbb{E}\left[X^2\right] = \sum_{i=1}^{\infty} i^2(1-p)^{(i-1)}p = \frac{p}{1-p}\frac{(1-p)(2-p)}{p^3},$$

then rearrange to get the expression for variance.

Bernoulli Random Variables

7.8. Write X for a Bernoulli random variable which takes the value 1 with probability p (and 0 with probability $(1-p)$).
(a) Show that $\mathbb{E}[X] = p$.
(b) Show that $\mathbb{E}\left[X^2\right] - \mathbb{E}[X]^2 = p(1-p)$

The Binomial Distribution

7.9. Show that $P_b(N-i; N, p) = P_b(i; N, p)$ for all i.

7.10. Write h_r for the number of heads obtained in r flips of a coin which has probability p of coming up heads. Compare the following two ways to compute the probability of getting i heads in five coin flips:

- Flip the coin three times, count h_3, then flip the coin twice, count h_2, then form $w = h_3 + h_2$.
- Flip the coin five times, and count h_5.

Show that the probability distribution for w is the same as the probability distribution for h_5. Do this by showing that

$$P(\{w = i\}) = \sum_{j=0}^{5} P(\{h_3 = j\} \cap \{h_2 = i - j\}) = P(\{h_5 = i\}).$$

7.11. Now we will do the previous exercise in a more general form. Again, write h_r for the number of heads obtained in r flips of a coin which has probability p of coming up heads. Compare the following two ways to compute the probability of getting i heads in N coin flips:

- Flip the coin t times, count h_t, then flip the coin $N - t$ times, count h_{N-t}, then form $w = h_t + h_{N-t}$.
- Flip the coin N times, and count h_N.

Show that the probability distribution for w is the same as the probability distribution for h_N. Do this by showing that

$$P(\{w = i\}) = \sum_{j=0}^{N} P(\{h_t = j\} \cap \{h_{N-t} = i - j\}) = P(\{h_N = i\}).$$

You will likely find the recurrence relation

$$P_b(i; N, p) = pP_b(i-1; N-1, p) + (1-p)P_b(i; N-1, p).$$

is useful.

7.12. An airline runs a regular flight with six seats on it. The airline sells six tickets. The gender of the passengers is unknown at time of sale, but women are as common as men in the population. All passengers always turn up for the flight. The pilot is eccentric, and will not fly a plane unless at least one passenger is female. What is the probability that the pilot flies?

7.13. An airline runs a regular flight with s seats on it. The airline always sells t tickets for this flight. The probability a passenger turns up for departure is p, and passengers do this independently. What is the probability that the plane travels with exactly 3 empty seats?

7.14. An airline runs a regular flight with s seats on it. The airline always sells t tickets for this flight. The probability a passenger turns up for departure is p, and passengers do this independently. What is the probability that the plane travels with 1 or more empty seats?

7.15. An airline runs a regular flight with 10 seats on it. The probability that a passenger turns up for the flight is 0.95. What is the smallest number of seats the airline should sell to ensure that the probability the flight is full (i.e. 10 or more passengers turn up) is bigger than 0.99? (you'll probably need to use a calculator or write a program for this).

The Multinomial Distribution

7.16. Show that the multinomial distribution

$$P_m(n_1, \ldots, n_k; N, p_1, \ldots, n_k) = \frac{N!}{n_1!n_2!\ldots n_k!} p_1^{n_1} p_2^{n_2} \ldots p_k^{n_k}$$

must satisfy the recurrence relation

$$\begin{aligned} P_m(n_1, \ldots, n_k; N, p_1, \ldots, p_k) &= p_1 P_m(n_1 - 1, \ldots, n_k; N - 1, p_1, \ldots, p_k) + \\ & \quad p_2 P_m(n_1, n_2 - 1, \ldots, n_k; N - 1, p_1, \ldots, p_k) + \ldots \\ & \quad p_k P_m(n_1, n_2, \ldots, n_k - 1; N - 1, p_1, \ldots, p_k) \end{aligned}$$

The Poisson Distribution

7.17. Compute the Taylor series for xe^x around $x = 0$. Use this and pattern matching to show that the mean of the Poisson distribution with intensity parameter λ is λ.

7.18. Compute the Taylor series for $(x^2 + x)e^x$ around $x = 0$. Use this and pattern matching to show that the variance of the Poisson distribution with intensity parameter λ is λ.

Sums of Continuous Random Variables

7.19. Write p_x for the probability density function of a continuous random variable X and p_y for the probability density function of a continuous random variable Y. Show that the probability density function of $S = X + Y$ is

$$p(s) = \int_{-\infty}^{\infty} p_x(s - u) p_y(u) du = \int_{-\infty}^{\infty} p_x(u) p_y(s - u) du$$

The Normal Distribution

7.20. Write

$$f(x) = \left(\frac{1}{\sqrt{2\pi}} \right) \exp\left(\frac{-x^2}{2} \right).$$

(a) Show that $f(x)$ is non-negative for all x.

(b) By integration, show that

$$\int_{-\infty}^{\infty} f(x) dx = 1,$$

so that $f(x)$ is a probability density function.

(c) Show that

$$\int_{-\infty}^{\infty} xf(x)dx = 0.$$

The easiest way to do this is to notice that $f(x) = f(-x)$

(d) Show that

$$\int_{-\infty}^{\infty} xf(x-\mu)dx = \mu.$$

The easiest way to do this is to change variables, and use the previous two exercises.

(e) Show that

$$\int_{-\infty}^{\infty} x^2 f(x)dx = 1.$$

You'll need to either do, or look up, the integral to do this exercise.

7.21. Write

$$g(x) = \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right]$$

Show that

$$\int_{-\infty}^{\infty} g(x)dx = \sqrt{2\pi}\sigma.$$

You can do this by a change of variable, and the results of the previous exercises.

7.22. Write

$$p(x) = \left(\frac{1}{\sqrt{2\pi}\sigma}\right)\exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right).$$

(a) Show that

$$\int_{-\infty}^{\infty} xp(x)dx = \mu$$

using the results of the previous exercises.

(b) Show that

$$\int_{-\infty}^{\infty} x^2 p(x)dx = \sigma$$

using the results of the previous exercises.

The Binomial Distribution for Large N

7.23. I flip a fair coin N times and count heads. We consider the probability that h, the fraction of heads, is in some range of numbers. *Hint:* If you know the range of numbers for h, you know the range for h/N.

(a) For $N = 1e6$, what is $P(\{h \in [49500, 50500]\})$?

(b) For $N = 1e4$, what is $P(\{h > 9000\})$?

(c) For $N = 1e2$, what is $P(\{h > 60\} \cup \{h < 40\})$?

CHAPTER 8

Markov Chains and Simulation

There are many situations where one must work with a sequence of random variables. For example, consider a bus queue; people arrive at random, and so do buses. What is the probability that the queue gets to a particular length? and what is the expected length of the queue? As another example, we could have a random model of purchases from a shop — under any particular rule for restoring inventory, what is the largest (resp. smallest) amount of stock on the shelf? what is the expected amount of stock on the shelf? It turns out that there is a simple and powerful model that applies to many sequences of random variables. One can often use this model to make closed-form predictions. In cases where closed-form predictions aren't available, one can use simulation methods to estimate probabilities and expectations.

8.1 MARKOV CHAINS

A Markov chain is a sequence of random variables which has important independence properties. We will describe these properties in some detail below; first, we give some examples. Markov chains are easily represented with the language of finite state machines. For some Markov chains, it is easy to determine probabilities and expectations of interest in closed form using simple methods. For others, it is tricky (straightforward, but unreasonable amounts of straightforward work). In these cases, we can estimate the relevant probabilities and expectations by simulating the finite state machine.

8.1.1 Motivating Example: Multiple Coin Flips

We start with three examples, each of which is easy to work. Each suggests a much harder question, however, which we need new machinery to handle.

Worked example 8.1 *Multiple Coin Flips - 1*

You choose to flip a fair coin until you see two heads in a row, and then stop. What is the probability that you flip the coin twice?

Solution: Because you stopped after two flips, you must have seen two heads. So $P(2 \text{ flips}) = P(\{HH\}) = P(\{H\})^2 = 1/4$.

Worked example 8.2 *Multiple Coin Flips - 2*

You choose to flip a fair coin until you see two heads in a row, and then stop. What is the probability that you flip the coin three times?

Solution: Because you stopped after three flips, you must have seen T, then H, then H; any other sequence either doesn't stop, or stops too early. We write this with the last flip last, so THH. So $P(3 \text{ flips}) = P(\{THH\}) = P(\{T\})P(\{H\})^2 = 1/8$.

Worked example 8.3 *Multiple Coin Flips - 3*

You choose to flip a fair coin until you see two heads in a row, and then stop. What is the probability that you flip the coin four times?

Solution: This is more interesting. The last three flips must have been THH (otherwise you'd go on too long, or end too early). But, because the second flip must be a T, the first could be either H or T. This means there are two sequences that work: $HTHH$ and $TTHH$. So $P(4 \text{ flips}) = 2/8 = 1/4$.

The harder question here is to ask what is $P(N)$, for N some number (larger than 4, because we know the answers in the other cases). It is unattractive to work this case by case (you could try this, if you don't believe me). One very helpful way to think about the coin flipping experiment is as a finite state machine (Figure 8.1). If you think of this machine as a conventional finite state machine, it accepts any string of T, H, that (a) ends with HH, and (b) has no other HH in it. Alternatively, you could think of this machine as encoding a probabilistic process. At each state, an event occurs with some probability (in this case, a coin comes up H or T, with probability $(1/2)$). The event causes the machine to follow the appropriate edge. If we take this view, this machine stops when we have flipped two heads in succession. It encodes our problem of computing the probability of flipping a coin N times then stopping — this is just the probability that the machine hits the end state after N transitions.

It turns out to be straightforward to construct a recurrence relation for $P(N)$ (i.e. an equation for $P(N)$ in terms of $P(N-1)$, $P(N-2)$, and so on). This property is quite characteristic of repeated experiments. For this example, first, notice that $P(1) = 0$. Now imagine you have flipped the coin N times and stopped. We can assume that $N > 3$, because we know what happens for the other cases. The *last* three flips must have been THH. Write Σ_N for a sequence that is: (a) N flips long; (b) ends in HH; and (c) contains no other subsequence HH. Equivalently, this is a string accepted by the finite state machine of figure 8.1. We must have that any

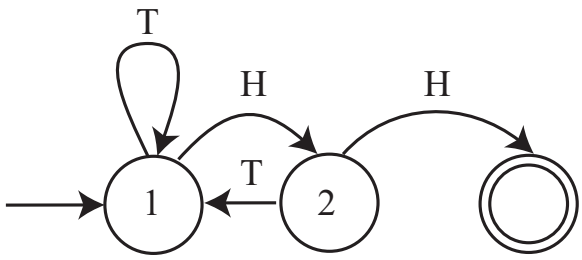

FIGURE 8.1: *A finite state machine representing the coin flip example. By convention, the end state is a double circle, and the start state has an incoming arrow. I've labelled the arrows with the event that leads to the transition, but haven't bothered to put in the probabilities, because each is 0.5.*

Σ_N has either the form $T\Sigma_{N-1}$ or the form $HT\Sigma_{N-2}$. But this means that

$$\begin{aligned} P(N) &= P(T)P(N-1) + P(HT)P(N-2) \\ &= (1/2)P(N-1) + (1/4)P(N-2) \end{aligned}$$

It is possible to solve this recurrence relation to get an explicit formula, but doing so would take us out of our way. You will check this recurrence relation in an exercise.

One really useful way to think of this recurrence relation is that represents an exercise in counting. We want to count all sequences that are of length N, and are accepted by the finite state machine of figure 8.1. This means they: (a) are N flips long; (b) end in HH; and (c) contain no other subsequence HH. Now work backward along the FSM. The only way to arrive at the final state is to be in state 1, then see HH. So you can obtain an acceptable N element sequence by (a) prepending a T to an acceptable $N-1$ element sequence or (b) prepending TH (which takes you to 2, then back to 1) to an acceptable $N-2$ element sequence. This line of reasoning can be made much more elaborate. There are a few examples in the exercises.

8.1.2 Motivating Example: The Gambler's Ruin

Another useful example is known as the **gambler's ruin**. Assume you bet \$1 a tossed coin will come up heads. If you win, you get \$1 and your original stake back. If you lose, you lose your stake. But this coin has the property that $P(H) = p < 1/2$. We will study what happens when you bet repeatedly.

Assume you have $\$s$ when you start. You will keep betting until either (a) you have \$0 (you are ruined; you can't borrow money) or (b) the amount of money you have accumulated is $\$j$, where $j > s$. The coin tosses are independent. We will compute $P(\text{ruined, starting with s}|p)$ the probability that you leave the table with nothing, when you start with $\$s$. For brevity, we write $P(\text{ruined, starting with s}|p) = p_s$. You can represent the gambler's ruin problem with a finite state machine as well. Figure 8.2 shows a representation of the gambler's ruin problem.

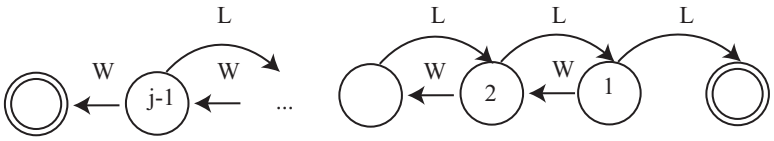

FIGURE 8.2: *A finite state machine representing the gambler's ruin example. I have labelled each state with the amount of money the gambler has at that state. There are two end states, where the gambler has zero (is ruined), or has j and decides to leave the table. The problem we discuss is to compute the probability of being ruined, given the start state is s. This means that any state except the end states could be a start state. I have labelled the state transitions with "W" (for win) and "L" for lose, but have omitted the probabilities.*

Worked example 8.4 *The gambler's ruin - 1*

Using the notation above, determine p_0 and p_j

Solution: We must have $p_0 = 1$, because if you have \$0, you leave the table. Similarly, if you have $\$j$, you leave the table with $\$j$, so you don't leave the table with nothing, so $p_j = 0$.

Worked example 8.5 *The gambler's ruin - 2*

Using the notation above, write a recurrence relation for p_s (the probability that you leave the table with nothing when you started with $\$s$.

Solution: Assume that you win the first bet. Then you have $\$s + 1$, so your probability of leaving the table with nothing now becomes p_{s+1}. If you lose the first bet, then you have $\$s - 1$, so your probability of leaving the table with nothing now becomes p_{s-1}. The coin tosses are independent, so we can write

$$p_s = pp_{s+1} + (1-p)p_{s-1}.$$

Some fairly lively work with series, relegated to the end of the chapter as exercises, yields

$$p_s = \frac{\left(\frac{1-p}{p}\right)^j - \left(\frac{1-p}{p}\right)^s}{\left(\frac{1-p}{p}\right)^j - 1}.$$

This expression is quite informative. Notice that, if $p < 1/2$, then $(1-p)/p > 1$. This means that as $j \to \infty$, we have $p_s \to 1$. If you gamble repeatedly on an unfair

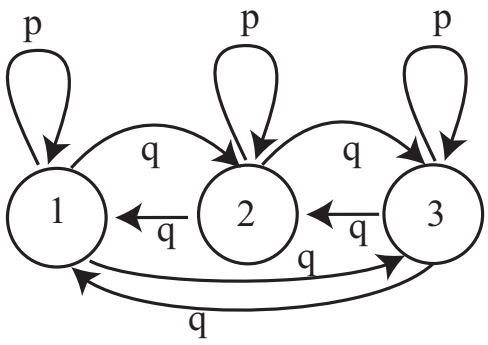

FIGURE 8.3: *A virus can exist in one of* 3 *strains. At the end of each year, the virus mutates. With probability* α*, it chooses uniformly and at random from one of the* 2 *other strains, and turns into that; with probability* $1 - \alpha$*, it stays in the strain it is in. For this figure, we have transition probabilities* $p = (1 - \alpha)$ *and* $q = (\alpha/2)$*.*

coin, the probability that you run out of money before you hit some threshold ($\$j$ in this case) tends to one.

8.1.3 Motivating Example: A Virus

Problems represented with a finite state machine don't need to have an end state. As one example, (which I got from ACC Coolen's lecture notes), we have a virus that can exist in one of k strains. At the end of each year, the virus mutates. With probability α, it chooses uniformly and at random from one of the $k - 1$ other strains, and turns into that; with probability $1 - \alpha$, it stays in the strain it is in (Figure 8.3 shows an example with three strains). For that figure, we have $p = (1 - \alpha)$ and $q = (\alpha/2)$. The virus just keeps on changing, and there is no end state. But there are a variety of very interesting questions we can try to answer. We would like to know, for example, the expected time to see a strain a second time, i.e. if the virus is in strain 1 at time 1, what is the expected time before it is in strain 1 again? If the virus has mutated many times, what is the probability that it is in each strain? and do these probabilities depend on the start strain?

8.1.4 Markov Chains

In Figure 8.1 and Figure 8.2, I showed the event that caused the state transitions. It is more usual to write a probability on the figure, as I did in Figure 8.3, because the probability of a state transition (rather than what caused it) is what really matters. The underlying object is now a weighted directed graph, because we have removed the events and replaced them with probabilities. These probabilities are known as **transition probabilities**; notice that the sum of transition probabilities over *outgoing* arrows must be 1.

We can now think of our process as a **biased random walk** on a weighted directed graph. A bug (or any other small object you prefer) sits on one of the graph's nodes. At each time step, the bug chooses one of the outgoing edges at

random. The probability of choosing an edge is given by the probabilities on the drawing of the graph (equivalently, the transition probabilities). The bug then follows that edge. The bug keeps doing this until it hits an end state.

This bug produces a sequence of random variables. If there are k states in the finite state machine, we can label each state with a number, $1 \ldots k$. At the n'th time step, the state of the process — the node that the bug is sitting on — is a random variable, which we write X_n. These random variables have an important property. The probability that X_n takes some particular value depends only on X_{n-1}, and not on any other previous state. If we know where the bug is at step $n-1$ in our model, we know where it could go, and the probability of each transition. Where it was at previous times does not affect this, *as long as we know its state at step* $n-1$.

A sequence of random variables X_n is a **Markov chain** if it has the property that, $P(X_n = j|\text{values of all previous states}) = P(X_n = j|X_{n-1})$, or, equivalently, only the last state matters in determining the probability of the current state. The probabilities $P(X_n = j|X_{n-1} = i)$ are the **transition probabilities**. Any model built by taking the transitions of a finite state machine and labelling them with probabilities must be a Markov chain. However, this is not the only way to build or represent a Markov chain.

One representation of a Markov chain uses a matrix of transition probabilities. We define the matrix $\mathcal{P}$ with $p_{ij} = P(X_n = j|X_{n-1} = i)$. Notice that this matrix has the properties that $p_{ij} \geq 0$ and

$$\sum_j p_{ij} = 1$$

because at the end of each time step the model must be in some state. Equivalently, the sum of transition probabilities for outgoing arrows is one. Non-negative matrices with this property are **stochastic matrices**. By the way, you should look very carefully at the i's and j's here — Markov chains are usually written in terms of *row* vectors, and this choice makes sense in that context.

Worked example 8.6 *Viruses*

Write out the transition probability matrix for the virus of Figure 8.3, assuming that $\alpha = 0.2$.

Solution: We have $P(X_n = 1|X_{n-1} = 1) = (1-\alpha) = 0.8$, and $P(X_n = 2|X_{n-1} = 1) = \alpha/2 = P(X_n = 3|X_{n-1} = 1)$; so we get

$$\begin{pmatrix} 0.8 & 0.1 & 0.1 \\ 0.1 & 0.8 & 0.1 \\ 0.1 & 0.1 & 0.8 \end{pmatrix}$$

Now imagine we do not know the initial state of the chain, but instead have a probability distribution. This gives $P(X_0 = i)$ for each state i. It is usual to take these k probabilities and place them in a k-dimensional row vector, which is

usually written π. For example, we might not know what the initial strain of the virus is, but just that each strain is equally likely. So for a 3-strain virus, we would have $\pi = [1/3, 1/3, 1/3]$. From this information, we can compute the probability distribution over the states at time 1 by

$$\begin{aligned} P(X_1 = j) &= \sum_i P(X_1 = j, X_0 = i) \\ &= \sum_i P(X_1 = j | X_0 = i) P(X_0 = i) \\ &= \sum_i p_{ij} \pi_i. \end{aligned}$$

If we write $\mathbf{p}^{(n)}$ for the row vector representing the probability distribution of the state at step n, we can write this expression as

$$\mathbf{p}^{(1)} = \pi \mathcal{P}.$$

Now notice that

$$\begin{aligned} P(X_2 = j) &= \sum_i P(X_2 = j, X_1 = i) \\ &= \sum_i P(X_2 = j | X_1 = i) P(X_1 = i) \\ &= \sum_i p_{ij} \left(\sum_{ki} p_{ki} \pi_k \right). \end{aligned}$$

so that

$$\mathbf{p}^{(n)} = \pi \mathcal{P}^n.$$

This expression is useful for simulation, and also allows us to deduce a variety of interesting properties of Markov chains.

Worked example 8.7 *Viruses*

We know that the virus of Figure 8.3 started in strain 1. After two state transitions, what is the distribution of states when $\alpha = 0.2$? when $\alpha = 0.9$? What happens after 20 state transitions? If the virus starts in strain 2, what happens after 20 state transitions?

Solution: If the virus started in strain 1, then $\pi = [1, 0, 0]$. We must compute $\pi(\mathcal{P}(\alpha))^2$. This yields $[0.66, 0.17, 0.17]$ for the case $\alpha = 0.2$ and $[0.4150, 0.2925, 0.2925]$ for the case $\alpha = 0.9$. Notice that, because the virus with small α tends to stay in whatever state it is in, the distribution of states after two years is still quite peaked; when α is large, the distribution of states is quite uniform. After 20 transitions, we have $[0.3339, 0.3331, 0.3331]$ for the case $\alpha = 0.2$ and $[0.3333, 0.3333, 0.3333]$ for the case $\alpha = 0.9$; you will get similar numbers even if the virus starts in strain 2. After 20 transitions, the virus has largely "forgotten" what the initial state was.

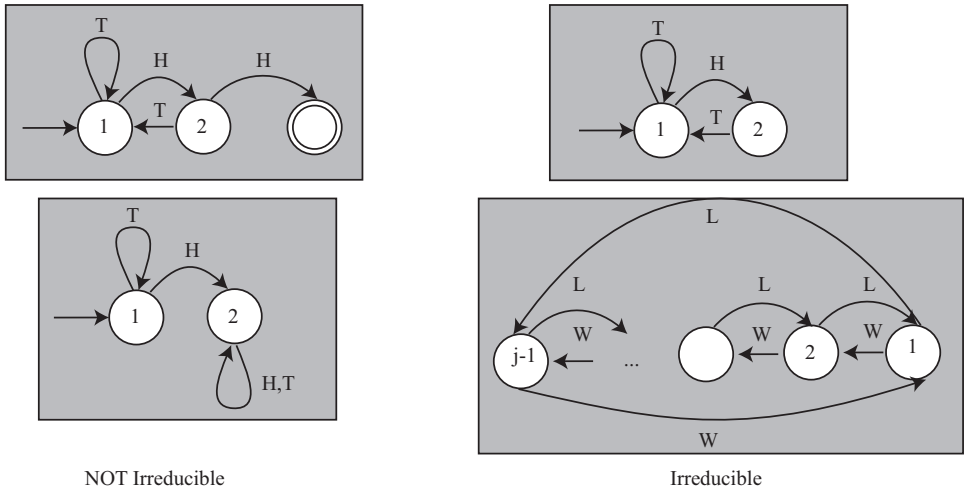

FIGURE 8.4: *Examples of finite state machines that give rise to Markov chains that are NOT irreducible (***left***) and irreducible (***right***). To obtain Markov chains from these drawings, we would have to give the probabilities of the events that lead to state transitions. We'll assume that none of the probabilities are zero; after that, the values don't matter for irreducibility analysis. The* **top left** *FSM is not irreducible, because it has an end state; once the bug reaches this point, it can't go anywhere else. The* **bottom left** *FSM is not irreducible, because the bug can get stuck in state 2.*

In example 7, the distribution of virus strains after a long interval appeared not to depend much on the initial strain. This property is true of many Markov chains. Assume that any state can be reached from any other state, by some sequence of transitions. Such chains are called **irreducible**; notice this means there is no end state, like the virus example. Irreducibility also means that the chain cannot get "stuck" in a state or a collection of states (Figure 8.4). Then there is a unique vector $\mathbf{s}$, usually referred to as the **stationary distribution**, such that for *any* initial state distribution π,

$$\lim_{n \to \infty} \pi \mathcal{P}^{(n)} = \mathbf{s}.$$

Equivalently, if the chain has run through many steps, it no longer matters what the initial distribution is. You expect that the probability distribution over states is $\mathbf{s}$.

8.1.5 Example: Particle Motion as a Markov Chain

One can find Markov chains in quite unexpected places, often with useful consequences. In this example, I will obtain a Markov chain without reasoning about graphs or finite state machines. We will investigate a particle moving under gravity, in 3 dimensions. Write the position of the particle as $\mathbf{p}(t)$, its velocity as $\mathbf{v}(t)$, its

acceleration as $\mathbf{a}(t)$, its mass as m, and the gravitational force as $\mathbf{g}$. Then we know that

$$\begin{aligned} \mathbf{v}(t) &= \frac{d\mathbf{p}}{dt} \\ \mathbf{a}(t) &= \frac{d\mathbf{v}}{dt} \\ &= \mathbf{g}. \end{aligned}$$

Now stack the position and the acceleration into a single vector $X(t) = (\mathbf{p}(t), \mathbf{v}(t))^T$. We could write these equations as

$$\frac{dX}{dt} = \mathcal{A}X + \mathbf{b}$$

where

$$\mathcal{A} = \begin{pmatrix} 0 & \mathcal{I} \\ 0 & 0 \end{pmatrix}$$

and

$$\mathbf{b} = \begin{pmatrix} 0 \\ \mathbf{g} \end{pmatrix}.$$

Now imagine that we look at the position, velocity and acceleration of the particle at fixed time instants, so we are really interested in $X_i = X(t_0 + i\Delta t)$. In this case, we can approximate $\frac{dX}{dt}$ by $(X_{i+1} - X_i)/\Delta t$. We then have

$$X_{i+1} = X_i + (\Delta t)\left(\mathcal{A}X_i + \mathbf{b}\right).$$

This is beginning to look like a Markov chain, because X_{i+1} depends only on X_i but not on any previous X. But there isn't any randomness here. We can fix that by assuming that the particle is moving in, say, a light turbulent wind. Then the acceleration at any time consists of (a) the acceleration of gravity and (b) a small, random force produced by the wind. Write $\mathbf{w}_i$ for this small, random force at time i, and $P(\mathbf{w}_i|i)$ for its probability distribution, which could reasonably depend on time. We can then rewrite our equation by writing

$$X_{i+1} = X_i + (\Delta t)\left(\mathcal{A}X_i + \mathbf{b}_r\right).$$

where

$$\mathbf{b}_r = \begin{pmatrix} 0 \\ \mathbf{g} + \mathbf{w}_i \end{pmatrix}.$$

Now the X_i are clearly random variables. We could get $P(X_{i+1}|X_i)$ by rearranging terms. If I know X_{i+1} and X_i, I know the value of $\mathbf{w}_i$; I could plug this into $P(\mathbf{w}_i|i)$ to yield $P(X_{i+1}|X_i)$.

Using a finite state machine in this example would be a bit unnatural. This example turns out to be surprisingly useful in applications, because (with other algorithmic machinery we can't go into here) it offers the basis for algorithms that can track moving objects by predicting where they will go next. Applications are widespread. Military applications include tracking aircraft, UFO's, missiles, etc. Civilian applications include surveillance in public places; games console interfaces that track moving people; and methods that can follow experimental mice as they move around their cages (useful for testing medicines).

8.2 SIMULATION

Many problems in probability can be worked out in closed form if one knows enough combinatorial mathematics, or can come up with the right trick. Textbooks are full of these, and we've seen some. Explicit formulas for probabilities are often extremely useful. But it isn't always easy or possible to find a formula for the probability of an event in a model. An alternative strategy is to build a simulation, run it many times, and count the fraction of outcomes where the event occurs. This is a simulation experiment.

8.2.1 Obtaining Uniform Random Numbers

Simulation is at its most useful when we try to estimate probabilities that are hard to calculate. Usually we have processes with some random component, and we want to estimate the probability of a particular outcome. To do so, we will need a supply of random numbers to simulate the random component. Here I will describe methods to get uniformly distributed random numbers, and Section 8.2.5 describes a variety of methods to get random numbers from other distributions.

I will describe features in Matlab, because I'm used to Matlab. It's a good programming environment for numerical work, particularly for working with matrices and vectors. You can expect pretty much any programming environment to provide a random number generator that returns a number uniformly distributed in the range $[0-1]$. If you know anything about representing numbers, you'll already have spotted something funny. The number that comes out of the random number generator must be represented by a small set of bits, but almost all numbers in the interval $[0-1]$ require an infinite number of bits to represent. We'll sweep this issue under the general carpet of floating point representations; it doesn't matter for anything we need to do. The Matlab function function to do this is called `rand`. It can also return matrices; for example, `rand(10, 20)` will get you a 10×20 table of independent uniformly distributed random numbers in the range $[0-1]$.

It is useful to get a uniformly distributed random integer in a particular range (for example, you might want to choose a random element in an array). You can do so with a random number generator and a function (like Matlab's `floor`) that returns the largest integer smaller than its argument. If I want a discrete random variable with uniform distribution, maximum value 100 and minimum value 7, I habitually choose a very tiny number (for this example, say $1e-7$) and do `floor((100-7-1e-7)*rand()+7)`. I use the $1e-7$ because I can never remember whether `rand` produces a number no larger than one, or one that is guaranteed to be smaller than one, and I never need to care about the very tiny differences in probability caused by the 1e-7. You might be able to do something cleaner if you bothered checking this point.

8.2.2 Computing Expectations with Simulations

Simulation is also a very good way to estimate expectations. Imagine we have a random variable X with probability distribution $P(X)$ that takes values in some domain D. Assume that we can easily produce independent simulations, and that we wish to know $\mathbb{E}[f]$, the expected value of the function f under the distribution

$P(X)$.

The weak law of large numbers tells us how to proceed. Define a new random variable $F = f(X)$. This has a probability distribution $P(F)$, which might be difficult to know. We want to estimate $\mathbb{E}[f]$, the expected value of the function f under the distribution $P(X)$. This is the same as $\mathbb{E}[F]$. Now if we have a set of IID samples of X, which we write x_i, then we can form a set of IID samples of F by forming $f(x_i) = f_i$. Write

$$F_N = \frac{\sum_{i=1}^{N} f_i}{N}.$$

This is a random variable, and the weak law of large numbers gives that, for any positive number ϵ

$$\lim_{N \to \infty} P(\{\| F_N - \mathbb{E}[F] \| > \epsilon\}) = 0.$$

You can interpret this as saying that, that for a set of IID random samples x_i, the probability that

$$\frac{\sum_{i=1}^{N} f(x_i)}{N}$$

is very close to $\mathbb{E}[f]$ is high for large N

Worked example 8.8 *Computing an Expectation*

Assume the random variable X is uniformly distributed in the range $[0-1]$, and the random variable Y is uniformly distributed in the range $[0-10]$. X and Z are independent. Write $Z = (Y - 5X)^3 - X^2$. What is $\mathsf{var}(\{Z\})$?

Solution: With enough work, one could probably work this out in closed form. An easy program will get a good estimate. We have that $\mathsf{var}(\{Z\}) = \mathbb{E}\left[Z^2\right] - \mathbb{E}[Z]^2$. My program computed 1000 values of Z (by drawing X and Y from the appropriate random number generator, then evaluating the function). I then computed $\mathbb{E}[Z]$ by averaging those values, and $\mathbb{E}[Z]^2$ by averaging their squares. For a run of my program, I got $\mathsf{var}(\{Z\}) = 2.76 \times 10^4$.

8.2.3 Computing Probabilities with Simulations

You can compute a probability using a simulation, too, because a probability can be computed by taking an expectation. Recall the property of indicator functions that

$$\mathbb{E}\left[\mathbb{I}_{[\mathcal{E}]}\right] = P(\mathcal{E})$$

(Section 6.2.5). This means that computing the probability of an event $\mathcal{E}$ involves writing a function that is 1 when the event occurs, and 0 otherwise; we then estimate the expected value of that function.

The weak law of large numers justifies this procedure. An experiment involves drawing a sample from the relevant probability distribution $P(X)$, then determining

whether event $\mathcal{E}$ occurred or not. We define a new random variable E, which takes the value 1 when the event $\mathcal{E}$ occurs during our experiment (i.e. with probability $P(\mathcal{E})$) and 0 when it doesn't (i.e. with probability $P(\mathcal{E}^c)$). Our experiment yields a sample of this random variable. We perform N experiments yielding a set of IID samples of this distribution e_i and compute $E_N = \frac{\sum_{i=1}^{N} e_i}{N}$. From the weak law of large numbers, we have that for any $\epsilon > 0$,

$$\lim_{N \to \infty} P(\{\| E_N - \mathbb{E}[E] \| \geq \epsilon\}) = 0$$

meaning that for large enough N, E_N will be very close to $\mathbb{E}[E] = P(\mathcal{E})$.

Worked example 8.9 *Computing a Probability for Multiple Coin Flips*

You flip a fair coin three times. Use a simulation to estimate the probability that you see three H's.

Solution: You really should be able to work this out in closed form. But it's amusing to check with a simulation. I wrote a simple program that obtained a 1000x3 table of uniformly distributed random numbers in the range $[0-1]$. For each number, if it was greater than 0.5 I recorded an H and if it was smaller, I recorded a T. Then I counted the number of rows that had 3 H's (i.e. the expected value of the relevant indicator function). This yielded the estimate 0.127, which compares well to the right answer.

Worked example 8.10 *Computing a Probability*

Assume the random variable X is uniformly distributed in the range $[0-1]$, and the random variable Y is uniformly distributed in the range $[0-10]$. Write $Z = (Y - 5X)^3 - X^2$. What is $P(\{Z > 3\})$?

Solution: With enough work, one could probably work this out in closed form. An easy program will get a good estimate. My program computed 1000 values of Z (by drawing X and Y from the appropriate random number generator, then evaluating the function) and counted the fraction of Z values that was greater than 3 (which is the relevant indicator function). For a run of my program, I got $P(\{Z > 3\}) \approx 0.619$

8.2.4 Simulation Results as Random Variables

The estimate of a probability or of an expectation that comes out of a simulation experiment is a random variable, because it is a function of random numbers. If you run the simulation again, you'll get a different value (unless you did something silly with the random number generator). Generally, you should expect this random variable to behave like a normal random variable. You can check this by constructing a histogram over a large number of runs. The mean of this random

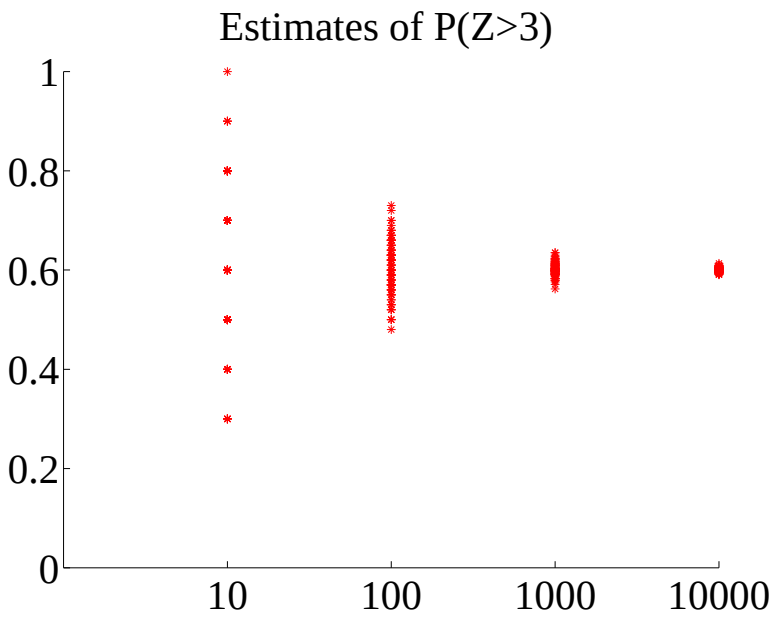

FIGURE 8.5: *Estimates of the probability from example 10, obtained from different runs of my simulator using different numbers of samples. In each case, I used 100 runs; the number of samples is shown on the horizontal axis. You should notice that the estimate varies pretty widely when there are only 10 samples, but the variance (equivalently, the size of the spread) goes down sharply as the number of samples increases to 1000. Because we expect these estimates to be roughly normally distributed, the variance gives a good idea of how accurate the original probability estimate is.*

variable is the parameter you are trying to estimate. It is useful to know that this random variable tends to be normal, because it means the standard deviation of the random variable tells you a lot about the likely values you will observe.

Another helpful rule of thumb, which is almost always right, is that the standard deviation of this random variable behaves like

$$\frac{C}{\sqrt{N}}$$

where C is a constant that depends on the problem and can be very hard to evaluate, and N is the number of runs of the simulation. What this means is that if you want to (say) double the accuracy of your estimate of the probability or the expectation, you have to run four times as many simulations. Very accurate estimates are tough to get, because they require immense numbers of simulation runs.

Figure 8.5 shows how the result of a simulation behaves when the number of runs changes. I used the simulation of example 10, and ran multiple experiments for each of a number of different samples (i.e. 100 experiments using 10 samples; 100 using 100 samples; and so on).

Small probabilities can be rather hard to estimate, as we would expect. In the case of example 10, let us estimate $P(\{Z > 950\})$. A few moments with a computer will show that this probability is of the order of 1e-3 to 1e-4. I obtained a million different simulated values of Z from my program, and saw 310 where $Z > 950$. This means that to know this probability to, say, three digits of numerical accuracy might involve a daunting number of samples. Notice that this does not contradict the rule of thumb that the standard deviation of the random variable defined by a simulation estimate behaves like $\frac{C}{\sqrt{N}}$; it's just that in this case, C is very large indeed.

8.2.5 Obtaining Random Samples

Building a successful simulation requires appropriate random numbers. Recall that anything we compute from a simulation can be thought of as an expectation (we used indicator functions to estimate probabilities). So we have some random variable X with distribution $P(X)$, a function f, and we wish to compute $\mathbb{E}[f]$, where the expectation is with respect to $P(X)$.

Most programming environments provide random number generators for uniform random numbers (I described the one for MATLAB briefly in Section 8.2.1) and for normally distributed random numbers. Building a really good, really fast random number generator is a major exercise. All sorts of tricks are involved, because it really matters to produce executable code that is as fast as possible on the target machine. This means that you don't have to build your own (and shouldn't, unless you can spend a lot of time and trouble on doing so).

Normal Random Variables

In pretty much any programming environment, you would also expect to find a random number generator that returns a normal random variable, with mean zero and standard deviation one. In Matlab, this function is called `randn`. Conveniently, `randn(3, 4)` will give you a 3×4 table of such numbers, which are independent of each other. As you would expect from section 1, to change the mean of this random number, you add a constant; to change the variance, you multiply by a constant. So in Matlab, if you want a normal random variable with mean 3 and standard deviation 4, you use `4*randn()+3`.

Rejection Sampling

Imagine you know a probability distribution describing a discrete random variable. In particular, you have one probability value for each of the numbers $1, 2, \ldots, N$ (all the others are zero). Write $p(i)$ for the probability of i. You can generate a sample of this distribution by the following procedure: first, generate a sample x of a uniform discrete random variable in the range $1, \ldots, N$; now generate a sample t of a uniformly distributed continuous random variable in the range $[0, 1]$; finally, if $t < p(x)$ report x, otherwise generate a new x and repeat. This process is known as **rejection sampling** (Algorithm 8.1).

To understand this procedure, it is best to think of it as a loop around a basic process. The basic process generates an x, then decides whether it is acceptable or not. The loop keeps invoking the basic process until it gets an acceptable x. Now

Listing 8.1: Matlab code for simple rejection sampling; this is inefficient, but simple to follow.

```
function rnum=rejectsample(pvec)
%
% pvec is a probability distribution over numbers
%  1, ..., size(pvec, 1)
%
nv=size(pvec, 1);
done=0;
while done==0
    ptr=floor(1+(nv-1e-10)*rand);
    % this gives me a uniform random
    % number in the range 1, .. nv
    pval=pvec(ptr);
    if rand<pval
        done=1;
        % i.e. accept
    end
end
rnum=ptr;
```

the probability that the basic process produces the value x_i, decides it is acceptable, and reports it is:

$$\begin{aligned} P(\{\text{report } x_i \text{ acceptable}\}) &= \begin{pmatrix} P(\{\text{accept } x_i\} \,|\, \{\text{generate } x_i\}) \\ \times \\ P(\{\text{generate } x_i\}) \end{pmatrix} \\ &= p(x_i) \times \frac{1}{N}. \end{aligned}$$

Now the loop keeps going until the basic process obtains an x_i that it decides is acceptable. This means the probability that the *loop* reports x_i is proportional to $p(x_i)$. But since $\sum_i p(x_i) = 1$, the probability that the *loop* reports x_i is *equal* to $p(x_i)$.

The problem here is that the basic process may not come up with an acceptable value x_i the first time; the loop might have to go on multiple times. If there are many x with small values of $p(x)$, we may find that it takes a very long time indeed to come up with a single sample. In the worst case, all values of $p(x)$ will be small. For example, for a uniform distribution on the range $1, \ldots, N$, $p(x)$ is $1/N$.

We can make things more efficient by noticing that multiplying the probability distribution by a constant doesn't change the relative frequencies with which numbers are selected. So this means that we can find the largest value $\hat{p}$ of $p(x)$, and form $q(x) = (1/\hat{p})p(x)$. Our process then becomes that shown in algorithm 8.2.

This process is not the most efficient available.

***** tree and point location algorithm

8.3 SIMULATION EXAMPLES

Computing probabilities and expectations from simulations should be so natural to a computer science student that it can be hard to see the magic. You estimate the

Listing 8.2: Matlab code for simple rejection sampling; this is somewhat more efficient.

```
function rnum=fasterrejectsample(pvec)
%
% pvec is a probability distribution over numbers
%  1, ..., size(pvec, 1)
%
nv=size(pvec, 1);
wv=max(pvec);
pv2=pvec/wv;  % we rescale
done=0;
while done==0
    ptr=floor(1+(nv-1e-10)*rand);
    % this gives me a uniform random
    % number in the range 1, .. nv
    pval=pv2(ptr); % work with rescaled probs
    if rand<pval
        done=1;
        % i.e. accept
    end
end
rnum=ptr;
```

probability of an event $\mathcal{E}$ by writing a program that runs N independent simulations of an experiment, counts how many times the event occurs (which we write $\#(\mathcal{E})$), and reports

$$\frac{\#(\mathcal{E})}{N}$$

as an estimate of $P(\mathcal{E})$. This estimate will not be exact, and may be different for different runs of the program. It's often a good, simple estimate.

Choosing N depends on a lot of considerations. Generally, a larger N means a more accurate answer, and also a slower program. If N is too small, you may find that you report 1 or 0 for the probability (think of what would happen if you measured $P(H)$ for a coin with one flip). One strategy is to run several simulations, report the mean, and use the standard deviation as some guide to the accuracy.

8.3.1 Simulating Experiments

Worked example 8.11 *Getting 14's with 20-sided dice*

You throw 3 fair 20-sided dice. Estimate the probability that the sum of the faces is 14 using a simulation. Use $N = [1e1, 1e2, 1e3, 1e4, 1e5, 1e6]$. Which estimate is likely to be more accurate, and why?

Solution: You need a fairly fast computer, or this will take a long time. I ran ten versions of each experiment for $N = [1e1, 1e2, 1e3, 1e4, 1e5, 1e6]$, yielding ten probability estimates for each N. These were different for each version of the experiment, because the simulations are random. I got means of $[0, 0.0030, 0.0096, 0.0100, 0.0096, 0.0098]$, and standard deviations of [00.00670.00330.00090.00020.0001]. This suggests the true value is around 0.0098, and the estimate from $N = 1e6$ is best. The reason that the estimate with $N = 1e1$ is 0 is that the probability is very small, so you don't usually observe this case at all in only ten trials.

Worked example 8.12 *Comparing simulation with computation*

You throw 3 fair six-sided dice. You wish to know the probability the sum is 3. Compare the true value of this probability with estimates from six runs of a simulation using $N = 10000$. What conclusions do you draw?

Solution: I ran six simulations with $N = 10000$, and got $[0.0038, 0.0038, 0.0053, 0.0041, 0.0056, 0.0049]$. The mean is 0.00458, and the standard deviation is 0.0007, which suggests the estimate isn't that great, but the right answer should be in the range $[0.00388, 0.00528]$ with high probability. The true value is $1/216 \approx 0.00463$. The estimate is tolerable, but not super accurate.

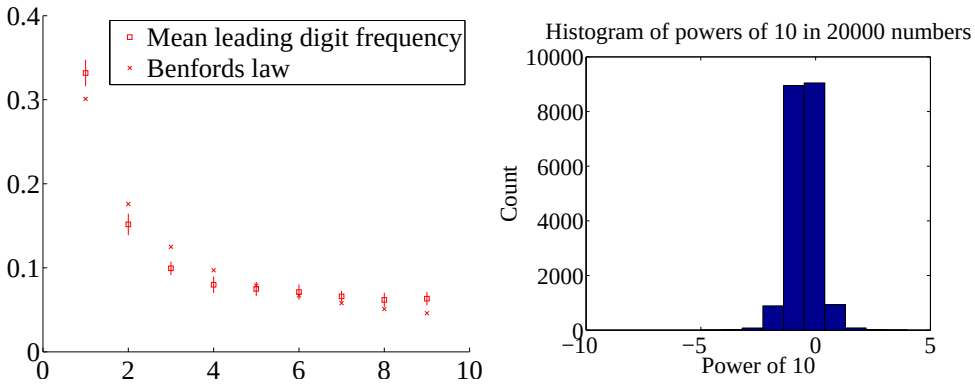

FIGURE 8.6: **Left** *summarizes results of a simulation where, for 20 runs, I generated 1000 pairs of independent random numbers uniformly distributed in* $[0-1]$ *and divided one by the other. The boxes show the mean over 20 runs, and the vertical bars show one standard deviation up and down. The crosses are predictions from Benford's law. Numbers generated in this way have a wide range of orders magnitude, so a conventional histogram isn't easy to plot. On the* **Right**, *a histogram of the first digit of the logarithm (i.e. the power of 10, or order of magnitude) of these numbers. Notice the smallest number is eight orders of magnitude smaller than the biggest.*

Worked example 8.13 *The First Digit of a Random Number*

We have seen (section **??**) that adding random variables tends to produce a normal random variable. What happens if one divides one random number by another? Solving this in closed form is tricky, but simulation gives some insight.

Solution: I wrote a short program that produced 20 experiments, each consisting of 1000 numbers. I obtained the numbers by generating two uniform random numbers in the range $[0-1]$, then dividing one by the other. Notice that doing so can produce both very big and very small numbers, so that plotting a histogram is tricky — most boxes will be empty unless we produce an immense quantity of data. Instead, I plot a histogram of the leading digit of the number in Figure 8.6.

Figure 8.6 also shows the mean over all 20 experiments of the fraction of leading digits that is one, etc. together with one standard deviation error bars. The figure also shows a histogram of the rounded log (i.e. the power of 10) to give some idea of the range of values one obtains. It turns out that there is a widely applicable law, called **Benford's law**, that predicts the frequency of the first digit of many types of random number. The main condition for Benford's law to apply is that the numbers cover a wide range of scales. The figure shows predictions from

Benford's law as well.

8.3.2 Simulating Markov Chains

Worked example 8.14 *Coin Flips with End Conditions*

I flip a coin repeatedly until I encounter a sequence HTHT, at which point I stop. What is the probability that I flip the coin nine times?

Solution: You might well be able to construct a closed form solution to this if you follow the details of example 1 and do quite a lot of extra work. A simulation is really straightforward to write; notice you can save time by not continuing to simulate coin flips once you've flipped past nine times. I got 0.0411 as the mean probability over 10 runs of a simulation of 1000 experiments each, with a standard deviation of 0.0056.

Worked example 8.15 *A Queue*

A bus is supposed to arrive at a bus stop every hour for 10 hours each day. The number of people who arrive to queue at the bus stop each hour has a Poisson distribution, with intensity 4. If the bus stops, everyone gets on the bus and the number of people in the queue becomes zero. However, with probability 0.1 the bus driver decides not to stop, in which case people decide to wait. If the queue is ever longer than 15, the waiting passengers will riot (and then immediately get dragged off by the police, so the queue length goes down to zero). What is the expected time between riots?

Solution: I'm not sure whether one could come up with a closed form solution to this problem. A simulation is completely straightforward to write. I get a mean time of 441 hours between riots, with a standard deviation of 391. It's interesting to play around with the parameters of this problem; a less conscientious bus driver, or a higher intensity arrival distribution, lead to much more regular riots.

Worked example 8.16 *Inventory*

A store needs to control its stock of an item. It can order stocks on Friday evenings, which will be delivered on Monday mornings. The store is old-fashioned, and open only on weekdays. On each weekday, a random number of customers comes in to buy the item. This number has a Poisson distribution, with intensity 4. If the item is present, the customer buys it, and the store makes $100; otherwise, the customer leaves. Each evening at closing, the store loses $10 for each unsold item on its shelves. The store's supplier insists that it order a fixed number k of items (i.e. the store must order k items each week). The store opens on a Monday with 20 items on the shelf. What k should the store use to maximise profits?

Solution: I'm not sure whether one could come up with a closed form solution to this problem, either. A simulation is completely straightforward to write. To choose k, you run the simulation with different k values to see what happens. I computed accumulated profits over 100 weeks for different k values, then ran the simulation five times to see which k was predicted. Results were $21, 19, 23, 20, 21$. I'd choose 21 based on this information.

For example 16, you should plot accumulated profits. If k is small, the store doesn't lose money by storing items, but it doesn't sell as much stuff as it could; if k is large, then it can fill any order but it loses money by having stock on the shelves. A little thought will convince you that k should be near 20, because that is the expected number of customers each week, so $k = 20$ means the store can expect to sell all its new stock. It may not be exactly 20, because it must depend a little on the balance between the profit in selling an item and the cost of storing it. For example, if the cost of storing items is very small compared to the profit, an very large k might be a good choice. If the cost of storage is sufficiently high, it might be better to never have anything on the shelves; this point explains the absence of small stores selling PC's.

8.3.3 Example: Ranking the Web by Simulating a Markov Chain

Perhaps the most valuable technical question of the last thirty years has been: Which web pages are interesting? Some idea of the importance of this question is that it was only really asked about 20 years ago, and at least one gigantic technology company has been spawned by a partial answer. This answer, due to Larry Page and Sergey Brin, and widely known as PageRank, starts with a Markov chain.

Figure 8.7 shows a picture of (a very small fraction of) the world wide web. I have drawn the web using a visual metaphor that should strongly suggest a finite state machine, and so a Markov chain. Each page is a state. Directed edges from page to page represent links. I count only the first link from a page to another page. Some pages are linked, others are not. I want to know how important each page is.

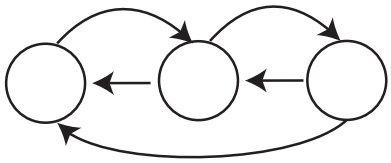

FIGURE 8.7: *A very small fraction of the web, drawn to suggest a finite state machine; each state represents a page, and each directed edge represents an outgoing link. A random web surfer could either (a) follow an outgoing link, chosen at random or (b) type in the URL of any page, chosen at random. Such a surfer would see lots of pages that have many incoming links from pages that have lots of incoming links, and so on. Pages like this are likely important, so that finding important pages is analogous to simulating a random web surfer.*

One way to think about importance is to think about what a random web surfer would do. The surfer can either (a) choose one of the outgoing links on a page at random, and follow it or (b) type in the URL of a new page, and go to that instead. As Figure 8.7 suggests, it is useful to think of this as a random walk on a finite state machine. We expect that this random surfer should see a lot of pages that have lots of incoming links from other pages that have lots of incoming links that (and so on). These pages are important, because lots of pages have linked to them.

For the moment, ignore the surfer's option to type in a URL. Write $r(i)$ for the importance of the i'th page. We model importance as leaking from page to page across outgoing links (the same way the surfer jumps). Page i receives importance down each incoming link. The amount of importance is proportional to the amount of importance at the other end of the link, and inversely proportional to the number of links leaving that page. So a page with only one outgoing link transfers all its importance down that link; and the way for a page to receive a lot of importance is for it to have a lot of important pages link to it alone. We write

$$r(j) = \sum_{i \to j} \frac{r(i)}{|i|}$$

where $|i|$ means the total number of links pointing *out* of page i. We can stack the $r(j)$ values into a *row* vector $\mathbf{r}$, and construct a matrix $\mathcal{P}$, where

$$p_{ij} = \begin{cases} \frac{1}{|i|} & \text{if } i \text{ points to } j \\ 0 & \text{otherwise} \end{cases}$$

With this notation, the importance vector has the property

$$\mathbf{r} = \mathbf{r}\mathcal{P}$$

and should look a bit like the stationary distribution of a random walk to you, except that $\mathcal{P}$ isn't stochastic — there may be some rows where the row sum of $\mathcal{P}$ is zero, because there are *no* outgoing links from that page. We can fix this easily

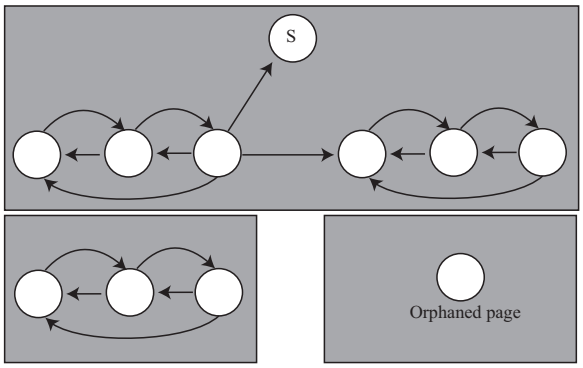

FIGURE 8.8: *The web isn't really like Figure 8.7. It's more like this figure (only bigger). In this figure, we see sections of the web that can't be reached by following links from other sections (the gray boxes); orphaned pages; and pages where a random walker would get stuck (S). Any algorithm for assigning importance must be able to deal with these effects.*

by replacing each row that sums to zero with $(1/n)\mathbf{1}$, where n is the total number of pages. Call the resulting matrix $\mathcal{G}$ (it's quite often called the **raw Google matrix**). Notice that doing this doesn't really change anything significant; pages with no outgoing links leak a tiny fraction of their importance to every other page.

Figure 8.7 isn't a particularly good model of the web (and not because it's tiny, either). Figure 8.8 shows some of the problems that occur. There are pages with no outgoing links (which we've dealt with), pages with no incoming links, and even pages with no links at all. Worse, a random walk can get trapped (in one of the gray boxes). One way to fix all this would be to construct a process that wandered around the web inserting links that clean up the structure of the graph. This strategy is completely infeasible, because the real web is much too big. Allowing the surfer to randomly enter a URL sorts out all of these problems, because it inserts an edge of small weight from every node to every other node. Now the random walk cannot get trapped.

There are a variety of possible choices for the weight of these inserted edges. The original choice was to make each inserted edge have the same weight. Write $\mathbf{1}$ for the n dimensional column vector containing a 1 in each component, and let $0 < \alpha < 1$. We can write the matrix of transition probabilities as

$$\mathcal{G}(\alpha) = \alpha\frac{(\mathbf{1}\mathbf{1}^T)}{n} + (1-\alpha)\mathcal{G}$$

where $\mathcal{G}$ is the original Google matrix. An alternative choice is to choose a weight for each web page, using anything from advertising revenues to page visit statistics to thaumaturgy (Google keeps quiet about the details). Write this weight vector $\mathbf{v}$, and require that $\mathbf{1}^T\mathbf{v} = 1$ (i.e. the coefficients sum to one). Then we could have

$$\mathcal{G}(\alpha, \mathbf{v}) = \alpha\frac{(\mathbf{1}\mathbf{v}^T)}{n} + (1-\alpha)\mathcal{G}.$$

Now the importance vector $\mathbf{r}$ is the (unique, though I won't prove this) *row* vector $\mathbf{r}$ such that

$$\mathbf{r} = \mathbf{r}\mathcal{G}(\alpha, \mathbf{v}).$$

How do we compute this vector? One natural algorithm is to start with some initial estimate, and propagate it. We write $\mathbf{r}^{(k)}$ for the estimated importance after the k'th step. We define updates by

$$(\mathbf{r}^{(k)}) = (\mathbf{r}^{(k-1)})\mathcal{G}(\alpha, \mathbf{v}).$$

We can't compute this directly, either, because $\mathcal{G}(\alpha, \mathbf{v})$ is unreasonably big so (a) we can't form or store $\mathcal{G}(\alpha, \mathbf{v})$ and (b) we can't multiply by it either. But we could estimate $\mathbf{r}$ with a random walk, because $\mathbf{r}$ is the stationary distribution of a Markov chain. If we simulate this walk for many steps, the probability that the simulation is in state j should be $r(j)$, at least approximately.

This simulation is easy to build. Imagine our random walking bug sits on a web page. At each time step, it transitions to a new page by either (a) picking from all existing pages at random, using $\mathbf{v}$ as a probability distribution on the pages (which it does with probability α); or (b) chooses one of the outgoing links uniformly and at random, and follows it (which it does with probability $1 - \alpha$). The stationary distribution of this random walk is $\mathbf{r}$. Another fact that I shall not prove is that, when α is sufficiently large, this random walk very quickly "forgets" it's initial distribution. As a result, you can estimate the importance of web pages by starting this random walk in a random location; letting it run for a bit; then stopping it, and collecting the page you stopped on. The pages you see like this are independent, identically distributed samples from $\mathbf{r}$; so the ones you see more often are more important, and the ones you see less often are less important.

8.3.4 Example: Simulating a Complicated Game

I will build several examples around a highly simplified version of a real card game. This game is Magic: The Gathering, and is protected by a variety of trademarks, etc. My version — MTGDAF — isn't very interesting as a game, but is simple enough to be studied, and interesting enough it casts some light on the real game. The game is played with decks of 60 cards. There are two types of card: Lands, and Spells. Lands can be placed on the play table and stay there permanently; Spells are played and then disappear. A Land on the table can be "tapped" or "untapped". Players take turns. Each player draws a hand of seven cards from a shuffled deck. In each turn, a player first untaps any Lands on the table, then draws a card, then plays a land onto the table (if the player has one in hand to play), then finally can play one or more spells. Each spell has a fixed cost (of $1, \ldots, 10$), and this cost is played by "tapping" a land (which is not untapped until the start of the next turn). This means that the player can cast only cheap spells in the early turns of the game, and expensive spells in the later turns.

Listing 8.3: Matlab code used to simulate the number of lands

```
simcards=[ones(24, 1); zeros(36, 1)]
% 1 if land, 0 otherwise
ninsim=10000;
nsims=10;
counts=zeros(nsims, 8);
for i=1:10
    for j=1:10000
        shuffle=randperm(60);
        hand=simcards(shuffle(1:7));
        %useful matlab trick here
        nlands=sum(hand);
        %ie number of lands
        counts(i, 1+nlands)=...
            counts(i, 1+nlands)+1;
        % number of lands could be zero
    end
end
probs=counts/ninsim;
mean(probs)
std(probs)
%%
```

Worked example 8.17 *MTGDAF — The number of lands*

Assume a deck of 60 cards has 24 Lands. It is properly shuffled, and you draw seven cards. You could draw $0, \ldots, 7$ Lands. Estimate the probability for each, using a simulation. Furthermore, estimate the error in your estimates.

Solution: The matlab function `randperm` produces a random permutation of given length. This means you can use it to simulate a shuffle of a deck, as in listing 8.3. I then drew 10, 000 random hands of seven cards, and counted how many times I got each number. Finally, to get an estimate of the error, I repeated this experiment 10 times and computed the standard deviation of each estimate of probability. This produced

0.0218 0.1215 0.2706 0.3082 0.1956 0.0686 0.0125 0.0012

for the probabilities (for 0 to 7, increasing number of lands to the right) and

0.0015 0.0037 0.0039 0.0058 0.0027 0.0032 0.0005 0.0004

for the standard deviations of these estimates.

Worked example 8.18 *MTGDAF — The number of lands*

What happens to the probability of getting different numbers of lands if you put only 15 Lands in a deck of 60? It is properly shuffled, and you draw seven cards. You could draw $0, \ldots, 7$ Lands. Estimate the probability for each, using a simulation. Furthermore, estimate the error in your estimates.

Solution: You can change one line in the listing to get

0.1159 0.3215 0.3308 0.1749 0.0489 0.0075 0.0006 0.0000

for the probabilities (for 0 to 7, increasing number of lands to the right) and

0.0034 0.0050 0.0054 0.0047 0.0019 0.0006 0.0003 0.0000

for the standard deviations of these estimates.

Worked example 8.19 *MTGDAF — Playing spells*

Assume you have a deck of 24 Lands, 10 Spells of cost 1, 10 Spells of cost 2, 10 Spells of cost 3, 2 Spells of cost 4, 2 Spells of cost 5, and 2 Spells of cost 6. Assume you always only play the cheapest spell in your hand (i.e. you never play two spells). What is the probability you will be able to play at least one spell on each of the first four turns?

Solution: This simulation requires just a little more care. You draw the hand, then simulate the first four turns. In each turn, you can only play a spell whose cost you can pay, and only if you have it. I used the matlab of listing 8.4 and listing 8.5; I found the probability to be 0.64 with standard deviation 0.01. Of course, my code might be wrong....

Listing 8.4: Matlab code used to simulate the four turns

```
simcards=[zeros(24, 1); ones(10, 1);...
    2*ones(10, 1);3*ones(10, 1); ...
    4*ones(2, 1); 5*ones(2, 1); 6*ones(2, 1)];
nsims=10;
ninsim=1000;
counts=zeros(nsims, 1);
for i=1:nsims
    for j=1:ninsim
        % draw a hand
        shuffle=randperm(60);
        hand=simcards(shuffle(1:7));
        %reorganize the hand
        cleanhand=zeros(7, 1);
        for k=1:7
            cleanhand(hand(k)+1)=cleanhand(hand(k)+1)+1;
            % ie count of lands, spells, by cost
        end
        landsontable=0;
        [playedspell1, landsontable, cleanhand]=...
            playround(landsontable, cleanhand, shuffle, ...
            simcards, 1);
        [playedspell2, landsontable, cleanhand]=...
            playround(landsontable, cleanhand, shuffle, ...
            simcards, 2);
        [playedspell3, landsontable, cleanhand]=...
            playround(landsontable, cleanhand, shuffle, ...
            simcards, 3);
        [playedspell4, landsontable, cleanhand]=...
            playround(landsontable, cleanhand, shuffle, ...
            simcards, 4);
        counts(i)=counts(i)+...
            playedspell1*playedspell2*...
            playedspell3*playedspell4;
    end
end
```

Worked example 8.20 *MTGDAF — Playing spells*

Now we use a different distribution of cards. Assume you have a deck of 20 Lands, 9 Spells of cost 1, 5 Spells of cost 2, 5 Spells of cost 3, 5 Spells of cost 4, 5 Spells of cost 5, and 11 Spells of cost 6. Assume you always only play the cheapest spell in your hand (i.e. you never play two spells). What is the probability you will be able to play at least one spell on each of the first four turns?

Solution: This simulation requires just a little more care. You draw the hand, then simulate the first four turns. In each turn, you can only play a spell whose cost you can pay, and only if you have it. I found the probability to be 0.33 with standard deviation 0.05. Of course, my code might be wrong....

Listing 8.5: Matlab code used to simulate playing a turn

```
function [playedspell, landsontable, cleanhand]=...
    playround(landsontable, cleanhand, shuffle, simcards, ...
    turn)
        % draw
ncard=simcards(shuffle(7+turn));
cleanhand(ncard+1)=cleanhand(ncard+1)+1;
% play land
if cleanhand(1)>0
     landsontable=landsontable+1;
     cleanhand(1)=cleanhand(1)-1;
end
playedspell=0;
if landsontable>0
    i=1; done=0;
    while done==0
        if cleanhand(i)>0
            cleanhand(i)=cleanhand(i)-1;
            playedspell=1;
            done=1;
        else
            i=i+1;
            if i>landsontable
                done=1;
            end
        end
    end
end
```

One engaging feature of the real game that is revealed by these very simple simulations is the tension between a players goals. The player would like to have few lands — so as to have lots of spells — but doing so means that there's a bigger chance of not being able to play a spell. Similarly, a player would like to have lots of powerful (=expensive) spells, but doing so means there's a bigger chance of not being able to play a spell. Players of the real game spend baffling amounts of time arguing with one another about the appropriate choices for a good set of cards.

Worked example 8.21 *MTGDAF — How long before you can play a spell of cost 3?*

Assume you have a deck of 15 Lands, 15 Spells of cost 1, 14 Spells of cost 2, 10 Spells of cost 3, 2 Spells of cost 4, 2 Spells of cost 5, and 2 Spells of cost 6. What is the expected number of turns before you can play a spell of cost 3? Assume you always play a land if you can.

Solution: I get 6.3, with a standard deviation of 0.1. The problem is it can take quite a large number of turns to get three lands out. I used the code of listings 8.6 and 8.7

Listing 8.6: Matlab code used to estimate number of turns before you can play a spell of cost 3

```
simcards=[zeros(15, 1); ones(15, 1);...
    2*ones(14, 1);3*ones(10, 1); ...
    4*ones(2, 1); 5*ones(2, 1); 6*ones(2, 1)];
nsims=10;
ninsim=1000;
counts=zeros(nsims, 1);
for i=1:nsims
    for j=1:ninsim
        % draw a hand
        shuffle=randperm(60);
        hand=simcards(shuffle(1:7));
        %reorganize the hand
        cleanhand=zeros(7, 1);
        for k=1:7
            cleanhand(hand(k)+1)=cleanhand(hand(k)+1)+1;
            % ie count of lands, spells, by cost
        end
        landsontable=0;
        k=0; played3spell=0;
        while played3spell==0;
            [played3spell, landsontable, cleanhand]=...
                play3round(landsontable, cleanhand, shuffle, ...
                simcards, k+1);
            k=k+1;
        end
        counts(i)=counts(i)+k;
    end
   counts(i)=counts(i)/ninsim;
end
```

Recall that you can reasonably expect that the probability you compute from a simulation behaves like normal data. Run a simulation experiment a large number of times and construct a data set whose entries are the probability from each experiment. This data should be normal. This means that, if you subtract the mean and divide by the standard deviation, you should get a histogram that looks like the standard normal curve. Figure 8.9 shows some examples of this effect. This effect is important, because it means that the right answer should be very few standard deviations away from the mean you compute — the standard deviation gives you quite a good idea of the accuracy of your estimate.

8.4 WHAT YOU SHOULD REMEMBER - NEED SOMETHING

PROBLEMS

8.1. Multiple coin flips:

(a) For example **??**, show that $P(5) = (3/32)$ by directly writing out the sequences of flips, and summing their probabilities.

(b) Now check the recurrence relation $P(N) = (1/2)P(N-1)+(1/4)P(N-2)$ for the case $N = 5$.

Listing 8.7: Matlab code used to simulate a turn to estimate the number of turns before you can play a spell of cost 3

```
function [played3spell, landsontable, cleanhand]=...
    play3round(landsontable, cleanhand, shuffle, simcards, ...
    turn)
        % draw
ncard=simcards(shuffle(7+turn));
cleanhand(ncard+1)=cleanhand(ncard+1)+1;
% play land
if cleanhand(1)>0
     landsontable=landsontable+1;
     cleanhand(1)=cleanhand(1)-1;
end
played3spell=0;
if (landsontable>=3)&&(cleanhand(4)>0)
    played3spell=1;
end
```

(c) What is $P(7)$?

8.2. Multiple die rolls: You roll a fair die until you see a 5, then a 6; after that, you stop. Write $P(N)$ for the probability that you roll the die N times.

(a) What is $P(1)$?

(b) Show that $P(2) = (1/36)$.

(c) Draw a finite state machine encoding all the sequences of die rolls that you could encounter. Don't write the events on the edges; instead, write their probabilities. There are 5 ways not to get a 5, but only one probability, so this simplifies the drawing.

(d) Show that $P(3) = (1/36)$.

(e) Now use your finite state machine to argue that $P(N) = (5/6)P(N-1) + (25/36)P(N-2)$.

8.3. More complicated multiple coin flips: You flip a fair coin until you see either HTH or THT, and then you stop. We will compute a recurrence relation for $P(N)$.

(a) Figure **??** shows a finite state machine. Check that this finite state machine represents all coin sequences that you will encounter.

(b) Write Σ_N for some string of length N accepted by this finite state machine. Use this finite state machine to argue that $Sigma_N$ has one of four forms:

1. $TT\Sigma_{N-2}$
2. $HH\Sigma_{N-3}$
3. $THH\Sigma_{N-2}$
4. $HTT\Sigma_{N-3}$

(c) Now use this argument to show that $P(N) = (1/2)P(N-2)+(1/4)P(N-3)$.

8.4. The gambler's ruin:

(a) Show that you can rearrange the recurrence relation of example 1 to get

$$p_{s+1} - p_s = \frac{(1-p)}{p}\left(p_s - p_{s-1}\right).$$

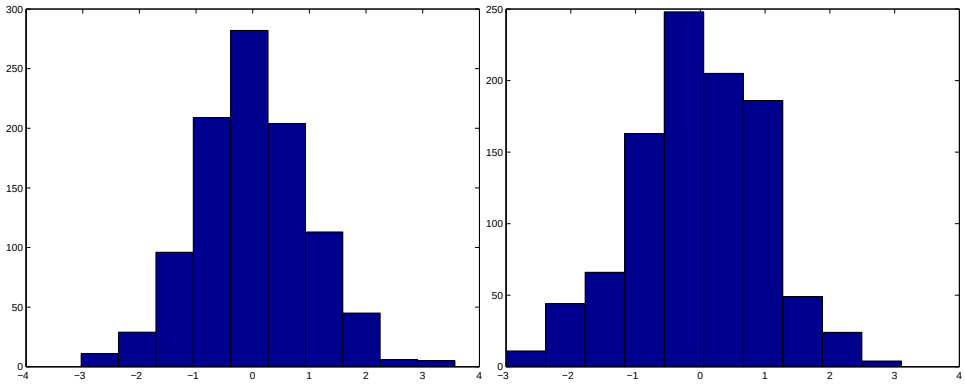

FIGURE 8.9: *Estimates of probabilities produced by simulation typically behave like normal data. On the* **left**, *I show a histogram of probabilities of having a hand of 3 Lands in the simulation of example 17; these are plotted in standard coordinates. On the* **right**, *I show a histogram of probability of playing a spell in each of the first four turns (example 20), from 1000 simulation experiments; again, these are plotted in standard coordinates. Compare these to the standard normal histograms of the previous chapter.*

Now show that this means that

$$p_{s+1} - p_s = \left(\frac{(1-p)}{p}\right)^2 (p_{s-1} - p_{s-2})$$

so that

$$\begin{aligned} p_{s+1} - p_s &= \left(\frac{(1-p)}{p}\right)^s (p_1 - p_0) \\ &= \left(\frac{(1-p)}{p}\right)^s (p_1 - 1)\,. \end{aligned}$$

(b) Now we need a simple result about series. Assume I have a series u_k, $k \geq 0$, with the property that

$$u_k - u_{k-1} = cr^{k-1}.$$

Show that

$$u_k - u_0 = c\left(\frac{r^k - 1}{r - 1}\right).$$

(c) Use the results of the last two steps to show that

$$p_s - 1 = (p_1 - 1)\left(\frac{\left(\frac{1-p}{p}\right)^s - 1}{\left(\frac{1-p}{p}\right) - 1}\right)$$

(d) Use the fact that $p_j = 0$ and the result of the last exercise to show

$$(p_1 - 1) = \frac{-1}{\left(\frac{\left(\frac{1-p}{p}\right)^j - 1}{\left(\frac{1-p}{p}\right) - 1}\right)}.$$

(e) Use the results of the previous exercises to show that

$$p_s = \frac{\left(\frac{1-p}{p}\right)^j - \left(\frac{1-p}{p}\right)^s}{\left(\frac{1-p}{p}\right)^j - 1}.$$

CHAPTER 9

Inference: Making Point Estimates

Inference is the process of drawing conclusions from data. One form of inference is to estimate a number, or set of numbers, that describes a dataset. The result is known as a **point estimate**. An alternative is to estimate an interval within which a number lies, with some degree of certainty. Such estimates are known as **interval estimates**. Finally, one might wish to assess the extent to which a body of evidence supports rejecting an hypothesis — known as **hypothesis testing**. In this chapter, we deal with point estimates. In the following chapter, we deal with interval estimates and hypothesis testing, which require an understanding of point estimates. There are two, somewhat distinct, situations in which we could make point estimates.

In the first, we have a dataset $\{\mathbf{x}\}$, and a probability model we believe applies to that dataset. But we need to select appropriate values of the parameters to ensure that the model describes the data. For example, we might have a set of N coin flips which we believe to be independent and identically distributed. Of these, k flips came up H. We know that a binomial distribution with $p(H) = p$ is a good model — but what value of p should we use? Your intuition is likely to suggest using k/N, but we'd like a more robust procedure than guessing. We need an inference procedure to obtain the unknown parameter from the data. Notice that this will be an estimate, rather than the "true" value. As we shall see, there is more than one possible procedure to apply, depending to some extent on the problem. In some cases (section 9.1), we estimate parameter values based solely on data; in others (section 9.2), we are able to use prior information about the parameters to affect the estimate.

In the second situation, we want to know some property of a population. For example, we may wish to know the mean weight of a person, or the mean response of a mouse to a drug. It would be a stretch to describe this population with one of the probability models that we have seen. In principle, the number we want is not even necessarily random; in principle, we could measure everyone on the planet and average the weights. In practice, this doesn't make sense, for quite straightforward reasons. You can't really weigh every person, or dose every mouse, on the planet. Instead, to estimate this property, we obtain a sample (some people; some mice; etc.) of the population, and estimate the property from the sample. There is now an important problem. Different samples lead to different estimates of the property. We will arrange to have the sample drawn randomly from the population, so the sample we see represents the value of a set of random variables. If you followed the proof of the weak law of large numbers, you should suspect that the mean of this sample could be a good estimate of the population mean. This turns out to be the case (section 1). However, there is some random error in the estimate, and we can tell (on average) how large the error caused by random sampling could be.

9.1 ESTIMATING MODEL PARAMETERS WITH MAXIMUM LIKELIHOOD

Assume we have a dataset $\mathcal{D} = \{\mathbf{x}\}$, and a probability model we believe applies to that dataset. Generally, application logic suggests the type of model (i.e. normal probability density; Poisson probability; geometric probability; and so on). But usually, we do not know the parameters of the model — for example, the mean and standard deviation of a normal distribution; the intensity of a poisson distribution; and so on. Our model will be better or worse depending on how well we choose the parameters. We need a strategy to estimate the parameters of a model from a sample dataset. Notice how each of the following examples fits this pattern.

Example: 9.1 *Inferring p from repeated flips — binomial*

We could flip the coin N times, and count the number of heads k. We know that an appropriate probability model for a set of independent coin flips is the binomial model $P(k; N, p)$. But we do not know p, which is the parameter — we need a strategy to extract a value of p from the data.

Example: 9.2 *Inferring p from repeated flips — geometric*

We could flip the coin repeatedly until we see a head. We know that, in this case, the number of flips has the geometric distribution with parameter p. In this case, the data is a sequence of T's with a final H from the coin flips. There are N flips (or terms) and the last flip is a head. We know that an appropriate probability model is the geometric distribution $P_g(N; p)$. But we do not know p, which is the parameter — we need a strategy to extract a value of p from the data.

Example: 9.3 *Inferring the intensity of spam — poisson*

It is reasonable to assume that the number of spam emails one gets in an hour has a Poisson distribution. But what is the intensity parameter λ? We could count the number of spam emails that arrive in each of a set of distinct hours, giving a dataset of counts $\mathcal{D}$. We need a strategy to wrestle an estimate of λ from this dataset.

Example: 9.4 *Inferring the mean and standard deviation of normal data*

Imagine we know for some reason that our data is well described by a normal distribution. We could ask what is the mean and standard deviation of the normal distribution that best represents the data?

We can write that model as $P(\mathcal{D}|\theta)$, where θ are parameters of the probability mode. The model is conditioned on θ, because if we knew θ we could evaluate the model. The expression $P(\mathcal{D}|\theta)$ is known as the **likelihood** of the data, and is often written $\mathcal{L}(\theta)$ (or $\mathcal{L}(\theta; \mathcal{D})$ if you want to remember that data is involved). Notice that this is unlike our models to date. In chapter 1, we assumed that we knew θ, and could then use the model to assign a probability to a data item. Here we *know* the value of $\mathcal{D}$. The likelihood is a function of θ.

9.1.1 The Maximum Likelihood Principle

We need a "reasonable" procedure to choose a value of θ to report. One — and we stress this is not the only one — is the **maximum likelihood principle**. This says: Choose θ such that $\mathcal{L}(\theta) = P(\mathcal{D}|\theta)$ is maximised, as a function of θ.

For the examples we work with, the data will be **independent and identically distributed** or **IID**. This means that each data item is an idependently obtained sample from the same probability distribution (see section 8.2.5). In turn, this means that the likelihood is a product of terms, one for each data item, which we can write as

$$\mathcal{L}(\theta) = P(\mathcal{D}|\theta) = \prod_{i \in \text{dataset}} P(d_i|\theta).$$

It is traditional to write θ for any set of parameters that are unknown. There are two, distinct, important concepts we must work with. One is the unknown parameter(s), which we will write θ. The other is the *estimate* of the value of that parameter, which we will write $\hat{\theta}$. This estimate is the best we can do — it may not be the "true" value of the parameter.

Worked example 9.1 *Inferring $p(H)$ for a coin from flips using a binomial model*

In N independent coin flips, you observe k heads. Use the maximum likelihood principle to infer $p(H)$.

Solution: The coin has $\theta = p(H)$, which is the unknown parameter. We know that an appropriate probability model is the binomial model $P(k; N, \theta)$. We have that

$$\mathcal{L}(\theta) = P(\mathcal{D}|\theta) = P_b(k; N, \theta) = \left(\begin{array}{c} N \\ k \end{array} \right) \theta^k (1-\theta)^{(N-k)}$$

which is a function of θ — the unknown probability that a coin comes up heads; k and N are known. We must find the value of θ that maximizes this expression. Now the maximum occurs when

$$\frac{\partial \mathcal{L}(\theta)}{\partial \theta} = 0.$$

We have

$$\frac{\partial \mathcal{L}(\theta)}{\partial \theta} = \left(\begin{array}{c} N \\ k \end{array} \right) \left(k\theta^{k-1}(1-\theta)^{(N-k)} - \theta^k (N-k)(1-\theta)^{(n-k-1)} \right)$$

and this is zero when

$$k\theta^{k-1}(1-\theta)^{(N-k)} = \theta^k (N-k)(1-\theta)^{(N-k-1)}$$

so the maximum occurs when

$$k(1-\theta) = \theta(N-k).$$

This means the maximum likelihood estimate is

$$\hat{\theta} = \frac{k}{N}$$

which is what we guessed would happen, but now we know why that guess "makes sense".

Worked example 9.2 *Inferring $p(H)$ from coin flips using a geometric model*

You flip a coin N times, stopping when you see a head. Use the maximum likelihood principle to infer $p(H)$ for the coin.

Solution: The coin has $\theta = p(H)$, which is the unknown parameter. We know that an appropriate probability model is the geometric model $P_g(N;\theta)$. We have that

$$\mathcal{L}(\theta) = P(\mathcal{D}|\theta) = P_g(N;\theta) = (1-\theta)^{(N-1)}\theta$$

which is a function of θ — the unknown probability that a coin comes up heads; N is known. We must find the value of θ that maximizes this expression. Now the maximum occurs when

$$\frac{\partial \mathcal{L}(\theta)}{\partial \theta} = 0 = ((1-\theta)^{(N-1)} - (N-1)(1-\theta)^{(N-2)}\theta)$$

So the maximum likelihood estimate is

$$\hat{\theta} = \frac{1}{N}.$$

We didn't guess this.

Worked example 9.3 *Inferring die probabilities from multiple rolls and a multinomial distribution*

You throw a die N times, and see n_1 ones, ... and n_6 sixes. Write $p_1, \ldots, p_6$ for the probabilities that the die comes up one, ..., six. Use the maximum likelihood principle to estimate $p_1, \ldots, p_6$.

Solution: The data are n, $n_1, \ldots, n_6$. The parameters are $\theta = (p_1, \ldots, p_6)$. $P(\mathcal{D}|\theta)$ comes from the multinomial distribution. In particular,

$$\mathcal{L}(\theta) = P(\mathcal{D}|\theta) = \frac{n!}{n_1! \ldots n_6!} p_1^{n_1} p_2^{n_2} \ldots p_6^{n_6}$$

which is a function of $\theta = (p_1, \ldots, p_6)$. Now we want to maximize this function by choice of θ. Notice that we could do this by simply making all p_i very large — but this omits a fact, which is that $p_1+p_2+p_3+p_4+p_5+p_6 = 1$. So we substitute using $p_6 = 1 - p_1 - p_2 - p_3 - p_4 - p_5$ (there are other, neater, ways of dealing with this issue, but they take more background knowledge). At the maximum, we must have that for all i,

$$\frac{\partial \mathcal{L}(\theta)}{\partial p_i} = 0$$

which means that, for p_i, we must have

$$n_i p_i^{(n_i-1)}(1-p_1-p_2-p_3-p_4-p_5)^{n_6} - p_i^{n_i} n_6 (1-p_1-p_2-p_3-p_4-p_5)^{(n_6-1)} = 0$$

so that, for each p_i, we have

$$n_i(1 - p_1 - p_2 - p_3 - p_4 - p_5) - n_6 p_i = 0$$

or

$$\frac{p_i}{1 - p_1 - p_2 - p_3 - p_4 - p_5} = \frac{n_i}{n_6}.$$

You can check that this equation is solved by

$$\hat{\theta} = \frac{1}{(n_1 + n_2 + n_3 + n_4 + n_5 + n_6)}(n_1, n_2, n_3, n_4, n_5, n_6)$$

The logarithm is a monotonic function (i.e. if $x > 0$, $y > 0$, $x > y$, then $\log(x) > \log(y)$). This means that the values of θ that maximise the log-likelihood are the same as the values that maximise the likelihood. This observation is very useful, because it allows us to transform a product into a sum. The derivative of a product involves numerous terms; the derivative of a sum is easy to take. We have

$$\log P(\mathcal{D}|\theta) = \log \prod_{i \in \text{dataset}} P(d_i|\theta) = \sum_{i \in \text{dataset}} \log P(d_i|\theta)$$

and in some cases, $\log P(d_i|\theta)$ takes a convenient, easy form. The log-likelihood of a dataset under a model is a function of the unknown parameters, and you will often see it written as

$$\log \mathcal{L}(\theta) = \sum_{i \in \text{dataset}} \log P(d_i|\theta).$$

Worked example 9.4 *Poisson distributions*

You observe N intervals, each of the same, fixed length (in time, or space). You know that, in these intervals, events occur with a Poisson distribution (for example, you might be observing Prussian officers being kicked by horses, or telemarketer calls...). You know also that the intensity of the Poisson distribution is the same for each observation. The number of events you observe in the i'th interval is n_i. What is the intensity, λ?

Solution: The likelihood is

$$\mathcal{L}(\theta) = \prod_{i \in \text{intervals}} P(\{n_i \text{ events}\}|\theta) = \prod_{i \in \text{intervals}} \frac{\theta^{n_i} e^{-\theta}}{n_i!}.$$

It will be easier to work with logs. The log-likelihood is

$$\log \mathcal{L}(\theta) = \sum_i (n_i \log \theta - \theta - \log n_i!)$$

so that we must solve

$$\frac{\partial \log \mathcal{L}(\theta)}{\partial \theta} = \sum_i (\frac{n_i}{\theta} - 1) = 0$$

which yields a maximum likelihood estimate of

$$\hat{\theta} = \frac{\sum_i n_i}{N}$$

Worked example 9.5 *The intensity of swearing*

A famously sweary politician gives a talk. You listen to the talk, and for each of 30 intervals 1 minute long, you record the number of swearwords. You record this as a histogram (i.e. you count the number of intervals with zero swear words, with one, etc.). For the first 10 intervals, you see

no. of swear words	no. of intervals
0	4
1	2
2	2
3	1
4	0

and for the following 20 intervals, you see

no. of swear words	no. of intervals
0	9
1	6
2	3
3	2
4	1

Assume that the politician's use of swearwords is Poisson. What is the intensity using the first 10 intervals? the second 20 intervals? all the intervals? why are they different?

Solution: Use the expression from worked example 4 to find

$$\begin{aligned}\hat{\lambda}_{10} &= \frac{\text{total number of swearwords}}{\text{number of intervals}} \\ &= \frac{7}{10}\end{aligned}$$

$$\begin{aligned}\hat{\lambda}_{20} &= \frac{\text{total number of swearwords}}{\text{number of intervals}} \\ &= \frac{22}{20}\end{aligned}$$

$$\begin{aligned}\hat{\lambda}_{30} &= \frac{\text{total number of swearwords}}{\text{number of intervals}} \\ &= \frac{29}{30}.\end{aligned}$$

These are different because the maximum likelihood estimate is an *estimate* — we can't expect to recover the exact value from a dataset. Notice, however, that the estimates are quite close.

Worked example 9.6 *Normal distributions*

Assume we have $x_1, \ldots, x_N$ which are data that can be modelled with a normal distribution. Use the maximum likelihood principle to estimate the mean of that normal distribution.

Solution: The likelihood of a set of data values under the normal distribution with unknown mean θ and standard deviation σ is

$$\begin{aligned}\mathcal{L}(\theta) &= P(x_1, \ldots x_N|\theta, \sigma) \\ &= P(x_1|\theta, \sigma)P(x_2|\theta, \sigma)\ldots P(x_N|\theta, \sigma) \\ &= \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x_i-\theta)^2}{2\sigma^2}\right)\end{aligned}$$

and this expression is a moderate nuisance to work with. The log of the likelihood is

$$\log \mathcal{L}(\theta) = \left(\sum_{i=1}^{N} -\frac{(x_i-\theta)^2}{2\sigma^2}\right) + \quad \text{term not depending on } \theta.$$

We can find the maximum by differentiating wrt θ and setting to zero, which yields

$$\begin{aligned}\frac{\partial \log \mathcal{L}(\theta)}{\partial \theta} &= \sum_{i=1}^{N} \frac{2(x_i-\theta)}{2\sigma^2} \\ &= 0 \\ &= \frac{1}{\sigma^2}\left(\sum_{i=1}^{N} x_i - N\theta\right)\end{aligned}$$

so the maximum likelihood estimate is

$$\hat{\theta} = \frac{\sum_{i=1}^{N} x_i}{N}$$

which probably isn't all that surprising. Notice we did not have to pay attention to σ in this derivation — we did not assume it was known, it just doesn't do anything.

Worked example 9.7 *Normal distributions -II*

Assume we have $x_1, \ldots, x_N$ which are data that can be modelled with a normal distribution. Use the maximum likelihood principle to estimate the standard deviation of that normal distribution.

Solution: Now we have to write out the log of the likelihood in more detail. Write μ for the mean of the normal distribution and θ for the unknown standard deviation of the normal distribution. We get

$$\log \mathcal{L}(\theta) = \left(\sum_{i=1}^{N} -\frac{(x_i - \mu)^2}{2\theta^2} \right) - N \log \theta + \text{Term not depending on } \theta$$

We can find the maximum by differentiating wrt σ and setting to zero, which yields

$$\frac{\partial \log \mathcal{L}(\theta)}{\partial \theta} = \frac{-2}{\theta^3} \sum_{i=1}^{N} \frac{-(x_i - \theta)}{2} - \frac{N}{\theta} = 0$$

so the maximum likelihood estimate is

$$\hat{\theta} = \sqrt{\frac{\sum_{i=1}^{N} (x_i - \mu)^2}{N}}$$

which probably isn't all that surprising, either.

The maximum likelihood principle has a variety of neat properties we cannot expound. One worth knowing about is **consistency**; for our purposes, this means that the maximum likelihood estimate of parameters can be made arbitrarily close to the right answer by having a sufficiently large dataset.

Another is that, in some cases, you can make online estimates. Assume, rather than seeing N elements of a dataset in one go, you get to see each one once, and you cannot store them. Assume that this dataset is modelled as normal data. Write $\hat{\mu}_k$ for the maximum likelihood estimate of the mean based on data items $1 \ldots k$ (and $\hat{\sigma}_k$ for the maximum likelihood estimate of the standard deviation, etc.). Notice that

$$\hat{\mu}_{k+1} = \frac{(k\hat{\mu}_k) + x_{k+1}}{(k+1)}$$

and that

$$\hat{\sigma}_{k+1} = \sqrt{\frac{(k\hat{\sigma}_k^2) + (x_{k+1} - \hat{\mu}_{k+1})^2}{(k+1)}}$$

This means that you can incorporate new data into your estimate as it arrives without keeping all the data. This process of updating a representation of a dataset as new data arrives is known as **filtering**.

9.1.2 Cautions about Maximum Likelihood

Our examples suggest some difficulties could occur in inference. The first is that it might be hard to find the maximum of the likelihood exactly. There are strong numerical methods for maximizing functions, and these are very helpful, but even today there are likelihood functions where it is very hard to find the maximum.

The second is that small amounts of data can present nasty problems. There is a body of mathematics, well outside the scope of this book, that implies that for lots of data that is well described by our model, maximum likelihood will give an answer very close to the "right" answer. This doesn't apply to small datasets. For example, in the binomial case, if we have only one flip we will estimate p as either 1 or 0. We should find this report unconvincing. In the geometric case, with a fair coin, there is a probability 0.5 that we will perform the estimate and then report that the coin has $p = 1$. This should also worry you. As another example, if we throw a die only a few times, we could reasonably expect that, for some i, $n_i = 0$. This doesn't necessarily mean that $p_i = 0$, though that's what the maximum likelihood inference procedure will tell us.

This creates a very important technical problem — how can I estimate the probability of events that haven't occurred? This might seem like a slightly silly question to you, but it isn't. Solving this problem has really significant practical consequences. For example, a really important part of natural language processing involves estimating the probability of groups of three words. These groups are usually known as "trigrams". People typically know an awful lot of words (tens to hundreds of thousands, depending on what you mean by a word). This means that there are a tremendous number of trigrams, and you can expect that any real dataset lacks almost all of them, because it isn't big enough. Some are missing because they don't occur in real life, but others are not there simply because they are unusual (eg "Atom Heart Mother" actually occurs in real life, but you may not have seen it all that often). Modern speech recognition systems need to know how probable *every* trigram is. Worse, if a trigram is modelled as having zero probability and actually occurs, the system will make a mistake, so it is important to model all such events as having a very small, but not zero, probability.

In summary, the maximum likelihood estimate is useful, and is consistent with intuition, but small datasets present some worries because there is a real prospect that the best estimate is wrong in a way that presents problems.

9.2 INCORPORATING PRIORS WITH BAYESIAN INFERENCE

Sometimes when we wish to estimate parameters of a model we have prior information. For example, we may have good reason to believe that some parameter is close to some value. We would like to take this information into account when we estimate the model. One way to do so is to place a **prior probability distribution** $p(\theta)$ on the parameters θ. Then, rather than working with the likelihood $p(\mathcal{D}|\theta)$, we could apply Bayes' rule, and form the **posterior** $p(\theta|\mathcal{D})$. This posterior represents the probability that θ takes various values, given the data $\mathcal{D}$. Extracting information from the posterior is usually called **Bayesian inference**. A natural estimate of θ is the value that maximizes the posterior. This estimate is sometimes known as a **maximum a priori estimate** or **MAP estimate**.

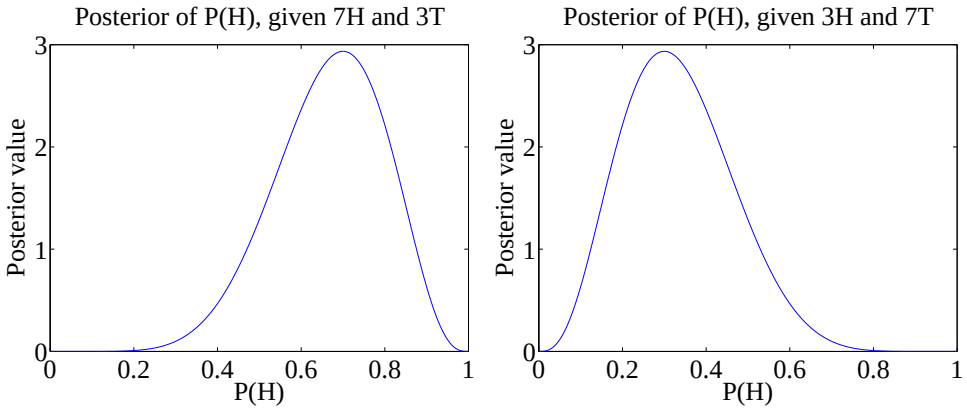

FIGURE 9.1: *The curves show a function* proportional to *the posterior on θ, for the two cases of example* **??**. *Notice that this information is rather richer than the single value we would get from maximum likelihood inference.*

9.2.1 Constructing the Posterior

Bayes' rule tells us that

$$p(\theta|\mathcal{D}) = \frac{P(\mathcal{D}|\theta)P(\theta)}{P(\mathcal{D})}$$

but (as we shall see) it can be hard to work out $P(\mathcal{D})$. For some problems, we might not need to know it.

Worked example 9.8 *Flipping a coin*

We have a coin with probability θ of coming up heads when flipped. We start knowing nothing about θ. We then flip the coin 10 times, and see 7 heads (and 3 tails). Plot a function proportional to $p(\theta|\{7 \text{ heads and } 3 \text{ tails}\})$. What happens if there are 3 heads and 7 tails?

Solution: We know nothing about p, except that $0 \leq \theta \leq 1$, so we choose a uniform prior on p. We have that $p(\{7 \text{ heads and } 3 \text{ tails}\}|\theta)$ is binomial. The joint distribution is $p(\{7 \text{ heads and } 3 \text{ tails}\}|\theta) \times p(\theta)$ but $p(\theta)$ is uniform, so doesn't depend on θ. So the posterior is *proportional* to: $\theta^7(1-\theta)^3$ which is graphed in figure 9.1. The figure also shows $\theta^3(1-\theta)^7$ which is *proportional* to the posterior for 3 heads and 7 tails. In each case, the evidence does not rule out the possibility that $\theta = 0.5$, but tends to discourage the conclusion. Maximum likelihood would give $\theta = 0.7$ or $\theta = 0.3$, respectively.

In Example 8, it is interesting to follow how the posterior on p changes as evidence come in, which is easy to do because the posterior is proportional to a binomial distribution. Figure 9.2 shows a set of these posteriors for different sets

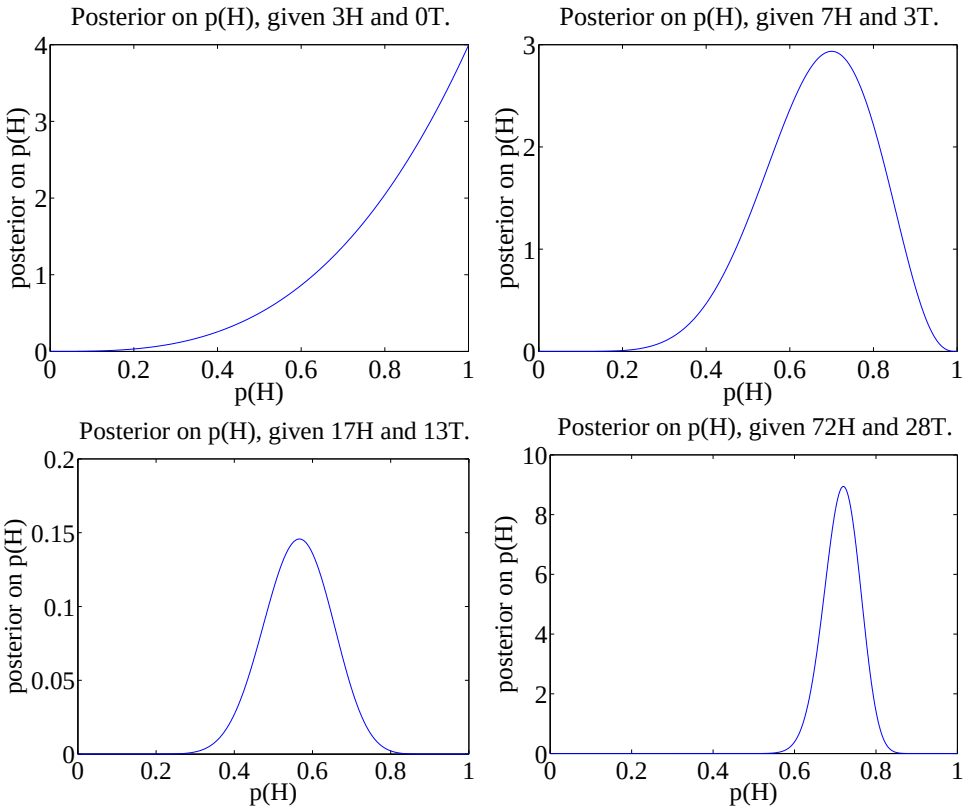

FIGURE 9.2: *The probability that an unknown coin will come up heads when flipped is $p(H)$. For these figures, I simulated coin flips from a coin with $p = 0.75$. I then plotted the posterior for various data. Notice how, as we see more flips, we get more confident about p.*

of evidence.

For other problems, we will need to marginalize out θ, by computing

$$P(\mathcal{D}) = \int_{\theta} P(\mathcal{D}|\theta)P(\theta)d\theta.$$

It is usually impossible to do this in closed form, so we would have to use a numerical integral. In some cases, $P(\theta)$ and $P(\mathcal{D}|\theta)$ are **conjugate**, meaning that $P(\theta|\mathcal{D})$ will take a familiar form and $P(\mathcal{D})$ follows easily.

Worked example 9.9 *Flipping a coin - II*

We have a coin with probability θ of coming up heads when flipped. We model the prior on θ with a Beta distribution, with parameters $\alpha > 0$, $\beta > 0$. We then flip the coin N times, and see h heads. What is $P(\theta|N, h, \alpha, \beta)$?

Solution: We have that $P(N, h|\theta)$ is binomial, and that $P(\theta|N, h, \alpha, \beta) \propto P(N, h|\theta)P(\theta|\alpha, \beta)$. This means that

$$P(\theta|N, h, \alpha, \beta) \propto \left(\begin{array}{c} N \\ h \end{array} \right) \theta^h (1-\theta)^{(N-h)} \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \theta^{(\alpha-1)} (1-\theta)^{(\beta-1)}.$$

and we can write

$$P(\theta|N, h, \alpha, \beta) \propto \theta^{(\alpha+h-1)} (1-\theta)^{(\beta+N-h-1)}.$$

Notice this has the form of a Beta distribution, so it is easy to recover the constant of proportionality. We have

$$P(\theta|N, h, \alpha, \beta) = \frac{\Gamma(\alpha+\beta+N)}{\Gamma(\alpha+h)\Gamma(\beta+N-h)} \theta^{(\alpha+h-1)} (1-\theta)^{(\beta+N-h-1)}.$$

Worked example 9.10 *More sweary politicians*

Example 5 gives some data from a sweary politician. Assume we have only the first 10 intervals of observations, and we wish to estimate the intensity using a Poisson model. Write θ for this parameter. Use a Gamma distribution as a prior, and write out the posterior.

Solution: We have that

$$p(\theta|\alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} \theta^{(\alpha-1)} e^{-\beta\theta} \text{ and } p(\mathcal{D}|\theta) = \frac{\theta^7 e^{-\theta}}{24}.$$

This means that

$$p(\theta|\mathcal{D}) \propto \theta^{(\alpha-1+7)} e^{-(\beta+1)\theta}.$$

Notice this has the form of another Gamma distribution, so we can write

$$p(\theta|\mathcal{D}) = \frac{(\beta+1)^{(\alpha+7)}}{\Gamma(\alpha+7)} \theta^{(\alpha-1+7)} e^{-(\beta+1)\theta}$$

9.2.2 The Posterior for Normal Data

There is a very useful construction for the posterior for data where the likelihood is normal. We start with a simple example. Assume we drop a measuring device down a borehole. It is designed to stop falling and catch onto the side of the hole after it has fallen μ_0 meters. On board is a device to measure its depth. This device reports a known constant times the correct depth plus a zero mean normal random variable, which we call "noise". The device reports depth every second.

The first question to ask is what depth do we believe the device is at *before* we receive any measurement? We designed the device to stop at μ_0 meters, so we are not completely ignorant about where it is. However, it may not have worked absolutely correctly. We choose to model the depth at which it stops as μ_0 meters plus a zero mean normal random variable. The second term could be caused by error in the braking system, etc. We could estimate the standard deviation of the second term (which we write σ_0) either by dropping devices down holes, then measuring with tape measures, or by analysis of likely errors in our braking system. The depth of the object is the unknown parameter of the model; we write this depth θ. Now the model says that θ is a normal random variable with mean μ_0 and standard deviation σ_0.

Notice that this model probably isn't exactly right — for example, there must be some probability in the model that the object falls beyond the bottom of the hole, which it can't do — but it captures some important properties of our system. The device should stop at or close to μ_0 meters most of the time, and it's unlikely to be too far away.

Now assume we receive a single measurement — what do we now know about the device's depth? The first thing to notice is that there is something to do here. Ignoring the prior and taking the measurement might not be wise. For example, imagine that the noise in the wireless system is large, so that the measurement is often corrupted — our original guess about the device's location might be better than the measurement. Write x_1 for the measurement. Notice that the scale of the measurement may not be the same as the scale of the depth, so the mean of the measurement is $c_1\theta$, where c_1 is a change of scale (for example, from inches to meters). We have that $p(x_1|\theta)$ is normal with mean $c_1\theta$ and standard deviation σ_{n1}. We would like to know $p(\theta|x_1)$.

We have that

$$\begin{aligned}\log p(\theta, x_1) &= \log p(x_1|\theta) + \log p(\theta) \\ &= -\frac{(x_1 - c_1\theta)^2}{2\sigma_{n1}^2} - \frac{(\theta - \mu_0)^2}{2\sigma_0^2} \\ &+ \text{terms not depending on } \theta \text{ or } x.\end{aligned}$$

We have two estimates of the position, θ, and we wish to come up with a representation of what we know about θ. One is x_1, which is a *measurement* — we know its value. The expected value of x_1 is $c_1\theta$, so we could infer θ from x_1. But we have another estimate of the position, which is μ_0. The posterior, $p(\theta|x_1)$, is a probability distribution on the variable θ; it depends on the known values x_1, μ_0, σ_0 and σ_{n1}. We need to determine its form. We can do so by some rearrangement of the expression for $\log p(\theta, x_1)$.

Notice first that this expression is of degree 2 in θ (i.e. it has terms θ^2, θ and things that don't depend on θ). This means that $p(\theta|x_1)$ must be a normal distribution, because we can rearrange its log into the form of the log of a normal distribution. This yields a fact of crucial importance.

Useful Fact: 9.1 *Normal distributions are conjugate*

A normal prior and a normal likelihood yield a normal posterior.

Write μ_1 for the mean of this distribution, and σ_{n1} for its standard deviation. The log of the distribution must be

$$-\frac{(\theta-\mu_1)^2}{2\sigma_1^2} + \text{terms not depending on } \theta.$$

The terms not depending on θ are not interesting, because if we know σ_1 those terms must add up to

$$\log\left(\frac{1}{\sqrt{2\pi}\sigma_1}\right)$$

so that the probability density function sums to one. Our goal is to rearrange terms into the form above. Notice that

$$-\frac{(\theta-\mu_1)^2}{2\sigma_p^2} = -\theta^2\left(\frac{1}{2\sigma_1^2}\right) + 2\theta\frac{\mu_1}{2\sigma_p^2} + \text{term not depending on } \theta$$

We have

$$\begin{aligned}
\log p(\theta|x_1) &= -\frac{(c_1\theta - x_1)^2}{2\sigma_{n1}^2} - \frac{(\theta-\mu_0)^2}{2\sigma_0^2} + \text{terms not depending on } \theta \\
&= -\theta^2\left(\frac{1}{2\left(\frac{\sigma_{n1}^2\sigma_0^2}{\sigma_{n1}^2+c_1^2\sigma_0^2}\right)}\right) + 2\theta\left(c_1\frac{x_1}{2\sigma_{n1}^2} + \frac{\mu_0}{2\sigma_0^2}\right) \\
&+ \quad \text{terms not depending on } \theta
\end{aligned}$$

which means that

$$\sigma_1^2 = \frac{\sigma_{n1}^2\sigma_0^2}{\sigma_{n1}^2 + c_1^2\sigma_0^2}$$

and

$$\begin{aligned}
\mu_1 &= 2\left(c_1\frac{x_1}{2\sigma_{n1}^2} + \frac{\mu_0}{2\sigma_0^2}\right)\frac{\sigma_{n1}^2\sigma_0^2}{\sigma_{n1}^2 + c_1^2\sigma_0^2} \\
&= \left(\frac{c_1x_1\sigma_0^2 + \mu_0\sigma_{n1}^2}{\sigma_{n1}^2\sigma_0^2}\right)\frac{\sigma_{n1}^2\sigma_0^2}{\sigma_{n1}^2 + c_1^2\sigma_0^2} \\
&= \frac{c_1x_1\sigma_0^2 + \mu_0\sigma_{n1}^2}{\sigma_{n1}^2 + c_1^2\sigma_0^2}.
\end{aligned}$$

These equations "make sense". Imagine that σ_0 is very small, and σ_{n1} is very big; then our new expected value of θ — which is μ_1 — is about μ_0. Equivalently, because our prior was very accurate, and the measurement was unreliable, our expected value is about the prior value. Similarly, if the measurement is reliable (i.e. σ_{n1} is small) and the prior has high variance (i.e. σ_0 is large), then our expected value of θ is about x_1/c_1 — i.e. the measurement, rescaled. I have put these equations, in a more general form, in a box below.

Useful Fact: 9.2 *Normal posteriors*

Assume we wish to estimate a parameter θ. The prior distribution for θ is normal, with known mean μ_π and known standard deviation σ_π. We receive a single data item x. The likelihood of this data item is normal with mean $c\theta$ and standard deviation σ_m, where c and σ_m are known. Then the posterior, $p(\theta|x, c, \sigma_m, \mu_\pi, \sigma_\pi)$, is normal, with mean

$$\frac{cx\sigma_\pi^2 + \mu_\pi\sigma_m^2}{\sigma_m^2 + c^2\sigma_\pi^2}$$

and standard deviation

$$\sqrt{\frac{\sigma_m^2\sigma_\pi^2}{\sigma_m^2 + c^2\sigma_\pi^2}}.$$

Assume a second measurement, x_2 arrives. We know that $p(x_2|\theta, c_2, \sigma_{n2})$ is normal with mean $c_2\theta$ and standard deviation σ_{n2}. In the example, we have a new measurement of depth — perhaps in a new, known, scale — with new noise (which might have larger, or smaller, standard deviation than the old noise) added. Then we can use $p(\theta|x_1, c_1, \sigma_{n1})$ as a prior to get a posterior $p(\theta|x_1, x_2, c_1, c_2, \sigma_{n1}, \sigma_{n2})$. Each is normal, by useful fact 1. Not only that, but we can easily obtain the expressions for the mean μ_2 and the standard deviation σ_2 *recursively* as functions of μ_1 and σ_1.

Applying useful fact 2, we have

$$\mu_2 = \frac{c_2x_2\sigma_1^2 + \mu_1\sigma_{n2}^2}{\sigma_{n2}^2 + c_2^2\sigma_1^2}$$

and

$$\sigma_2^2 = \frac{\sigma_{n2}^2\sigma_1^2}{\sigma_{n2}^2 + c_2^2\sigma_1^2}.$$

But what works for 2 and 1 will work for $k+1$ and k. We know the posterior after k measurements will be normal, with mean μ_k and standard deviation σ_k. The $k+1$'th measurement x_{k+1} arrives, and we have $p(x_{k+1}|\theta, c_{k+1}, \sigma_{n(k+1)})$ is normal. Then the posterior is normal, and we can write the mean μ_{k+1} and the standard deviation σ_{k+1} recursively as functions of μ_k and σ_k. The result is worth putting in a box.

Useful Fact: 9.3 *Online updating of normal posteriors*

Assume we wish to estimate a parameter θ. The prior distribution for θ is normal, with known mean μ_π and known standard deviation σ_π. All data is normal conditioned on θ. We have already received k data items. The posterior $p(\theta|x_1, \ldots, x_k, c_1, \ldots, c_k, \sigma_{n1}, \ldots, \sigma_{nk}, \mu_\pi, \sigma_\pi)$ is normal, with mean μ_k and standard deviation σ_k. We receive a new data item x_{k+1}. The likelihood of this data item is normal with mean $c\theta$ and standard deviation $\sigma_{n(k+1)}$, where c_{k+1} and $\sigma_{n(k+1)}$ are known. Then the posterior, $p(\theta|x_1, \ldots, x_{k+1}, c_1, \ldots, c_k, c_{k+1}, \sigma_{n1}, \ldots, \sigma_{n(k+1)}, \mu_\pi, \sigma_\pi)$, is normal, with mean

$$\mu_{k+1} = \frac{c_{k+1}x_{k+1}\sigma_k^2 + \mu_k\sigma_{n(k+1)}^2}{\sigma_{n(k+1)}^2 + c_{k+1}^2\sigma_k^2}$$

and

$$\sigma_{k+1}^2 = \frac{\sigma_{n(k+1)}^2\sigma_k^2}{\sigma_{n(k+1)}^2 + c_{k+1}^2\sigma_k^2}.$$

Again, notice the very useful fact that, if everything is normal, we can update our posterior representation when new data arrives using a very simple recursive form.

9.2.3 MAP Inference

Look at example 1, where we estimated the probability a coin would come up heads with maximum likelihood. We could not change our estimate just by knowing the coin was fair, but we could come up with a number for $\theta = p(H)$ (rather than, say, a posterior distribution). A natural way to produce a point estimate for θ that incorporates prior information is to choose $\hat{\theta}$ such that

$$\hat{\theta} = \underset{\theta}{\text{argmax}}\ P(\theta|\mathcal{D}) = \underset{\theta}{\text{argmax}}\ \frac{P(\theta, \mathcal{D})}{P(\mathcal{D})}$$

This is the MAP estimate. If we wish to perform MAP inference, $P(\mathcal{D})$ doesn't matter (it changes the value, but not the location, of the maximum). This means we can work with $P(\theta, \mathcal{D})$, often called the **joint** distribution.

Worked example 9.11 *Flipping a coin - II*

We have a coin with probability θ of coming up heads when flipped. We model the prior on θ with a Beta distribution, with parameters $\alpha > 0$, $\beta > 0$. We then flip the coin N times, and see h heads. What is the MAP estimate of θ?

Solution: We have that

$$P(\theta|N, h, \alpha, \beta) = \frac{\Gamma(\alpha + \beta + N)}{\Gamma(\alpha + h)\Gamma(\beta + N - h)}\theta^{(\alpha+h-1)}(1-\theta)^{(\beta+N-h-1)}.$$

You can get the MAP estimate by differentiating and setting to 0, yielding

$$\hat{\theta} = \frac{\alpha - 1 + h}{\alpha + \beta - 2 + N}.$$

This has rather a nice interpretation. You can see α and β as extra counts of heads (resp. tails) that are added to the observed counts. So, for example, if you were fairly sure that the coin should be fair, you might make α and β large and equal. When $\alpha = 1$ and $\beta = 1$, we have a uniform prior as in the previous examples.

Worked example 9.12 *More sweary politicians*

We observe our swearing politician for N intervals, seeing n_i swear words in the i'th interval. We model the swearing with a Poisson model. We wish to estimate the intensity, which we write θ. We use a Gamma distribution for the prior on θ. What is the MAP estimate of θ?

Solution: Write $T = \sum_{i=1}^{N}$. We have that

$$p(\theta|\mathcal{D}) = \frac{(\beta+1)^{(\alpha+T)}}{\Gamma(\alpha+T)}\theta^{(\alpha-1+T)}e^{-(\beta+1)\theta}$$

and the MAP estimate is

$$\hat{\theta} = \frac{(\alpha - 1 + T)}{(\beta + 1)}$$

(which you can get by differentiating with respect to θ, then setting to zero). Notice that if β is close to zero, you can interpret α as extra counts; if β is large, then it strongly discourages large values of $\hat{\theta}$, even if the counts are large.

Worked example 9.13 *Normal data*

Assume you see N datapoints x_i which are modelled by a normal distribution with unknown mean θ and with known standard deviation σ. You model the prior on θ using a normal distribution with mean μ_0 and standard deviation σ_0. What is the MAP estimate of the mean?

Solution: Recall that the maximum value of a normal distribution occurs at its mean. Now problem is covered by useful fact 2, but in this case we have $c_i = 1$ for each data point, and $\sigma_i = \sigma$. We can write

$$\mu_N = \frac{x_N \sigma_{N-1}^2 + \mu_{N-1}\sigma^2}{\sigma^2 + \sigma_{N-1}^2}$$

and

$$\sigma_N^2 = \frac{\sigma^2 \sigma_{N-1}^2}{\sigma^2 + \sigma_{N-1}^2}.$$

and evaluate the recursion down to μ_0, σ_0.

9.2.4 Cautions about Bayesian Inference

Just like maximum likelihood inference, bayesian inference is not a recipe that can be applied without thought. It turns out that, when there is a lot of data, the prior has little inference on the outcome of the inference, and the MAP solution looks a lot like the maximum likelihood solution. So the difference between the two approaches is most interesting when there is little data, where the prior matters. The difficulty is that it might be hard to know what to use as a good prior. In the examples, I emphasized mathematical convenience, choosing priors that lead to clean posteriors. There is no reason to believe that nature uses conjugate priors (even though conjugacy is a neat property). How should one choose a prior for a real problem?

This isn't an easy point. If there is little data, then the choice could really affect the inference. Sometimes we're lucky, and the logic of the problem dictates a choice of prior. Mostly, we have to choose and live with the consequences of the choice. Often, doing so is succesful in applications.

The fact we can't necessarily justify a choice of prior seems to be one of life's inconveniences, but it represents a significant philosophical problem. It's been at the core of a long series of protracted, often quite intense, arguments about the philosophical basis of statistics. I haven't followed these arguments closely enough to summarize them; they seem to have largely died down without any particular consensus being reached.

9.3 SAMPLES, URNS AND POPULATIONS

Very often the data we see is a small part of the data we could have seen, if we'd been able to collect enough data. We need to know how the measurements we make

on the dataset relate to the measurements we could have made, if we had all the data. This situation occurs very often. For example, imagine we wish to know the average weight of a rat. This isn't random; you could weigh every rat on the planet, and then average the answers. But doing so would absurd (among other things, you'd have to weigh them all at the same time, which would be tricky). Instead, we weight a small set of rats, chosen rather carefully. If we have chosen sufficiently carefully, then the answer from the small set is quite a good representation of the answer from the whole set.

The data we could have observed, if we could have seen everything, is the **population**. The data we actually have is the **sample**. We would like to know the mean of the population, but can see only the sample; surprisingly, we can say a great deal from the sample alone, assuming that it is chosen appropriately.

Assume we have a population $\{x\}$, for $i = 1, \ldots, N_p$. Notice the subscript here — this is the number of items in the population. The population could be unreasonably big: for example, it could consist of all the people in the world. We want to know the mean of this dataset, but we do not get to see the whole dataset. Instead, we see the sample.

9.3.1 Estimating the Population Mean from a Sample

How the sample is obtained is key to describing the population. We will focus on only one model (there are lots of others). In our model, the sample is obtained by choosing a fixed number of data items. Write k for the number of data items in the sample. We expect k is a lot smaller than N_p. Each item is chosen independently, and fairly. This means that each time we choose, we choose one from the entire set of N_p data items, and each has the same probability of being chosen. This is sometimes referred to as "sampling with replacement".

One natural way to think about sampling with replacement is to imagine the data items as being written on tickets, which are placed in an urn (old-fashioned word for a jar, now used mainly by statisticians and morticians). You obtain the sample by repeating the following experiment k times: shake the urn; take a ticket from the urn and write down the data on the ticket; put it back in the urn. Notice that, in this case, each sample is drawn from the same urn. This is important, and makes the analysis easier. If we had not put the ticket back, the urn would change between samples.

We summarize the whole dataset with its mean, which we write $\mathsf{popmean}(\{x\})$. This is known as the **population mean**. The notation is just to drive home the facts that it's the mean of the whole population, and that we don't, and can't, know it. The whole point of this exercise is to *estimate* this mean.

We would like to estimate the mean of the whole dataset from the items that we actually see. Imagine we draw k tickets from the urn as above, and average the values. The result is a random variable, because different draws of k tickets will give us different values. Write $X^{(k)}$ for this random variable, which is referred to as the **sample mean**. Because expectations are linear, we must have that

$$\mathbb{E}\left[X^{(k)}\right] = \frac{1}{k}\left(\mathbb{E}\left[X^{(1)}\right] + \ldots + \mathbb{E}\left[X^{(1)}\right]\right) = \mathbb{E}\left[X^{(1)}\right]$$

(where $X^{(1)}$ is the random variable whose value is obtained by drawing one ticket

from the urn). Now

$$\begin{aligned}
\mathbb{E}\left[X^{(1)}\right] &= \sum_{i \in 1,\ldots N_p} x_i p(i) \\
&= \sum_{i \in 1,\ldots N_p} x_i \frac{1}{N_p} \qquad \text{because we draw fairly from the urn} \\
&= \frac{\sum_{i \in 1,\ldots N_p} x_i}{N_p} \\
&= \mathsf{popmean}\left(\{x\}\right)
\end{aligned}$$

which is the mean value of the items in the urn. This means that

$$\mathbb{E}\left[X^{(k)}\right] = \mathsf{popmean}\left(\{x_i\}\right).$$

Under our sampling model, the expected value of the sample mean is the population mean.

Useful Facts: 9.4 *Sample means and population means*

The sample mean is a random variable. It is random, because different samples from the population will have different values of the sample mean. The expected value of this random variable is the population mean.

We will not get the same value of $X^{(k)}$ each time we perform the experiment, because we see different data items in each sample. So $X^{(k)}$ has variance, and this variance is important. If it is large, then each estimate is quite different. If it is small, then the estimates cluster. Knowing the variance of $X^{(k)}$ would tell us how accurate our estimate of the population mean is.

9.3.2 The Variance of the Sample Mean

We write $\mathsf{popsd}\left(\{x\}\right)$ for the standard deviation of the whole population of $\{x\}$. Again, we write it like this to keep track of the facts that (a) it's for the whole population and (b) we don't — and usually can't — know it.

We can compute the variance of $X^{(k)}$ (the sample mean) easily. We have

$$\mathsf{var}\left[X^{(k)}\right] = \mathbb{E}\left[(X^{(k)})^2\right] - \mathbb{E}\left[X^{(k)}\right]^2 = \mathbb{E}\left[(X^{(k)})^2\right] - \left(\mathsf{popmean}\left(\{x\}\right)\right)^2$$

so we need to know $\mathbb{E}\left[(X^{(k)})^2\right]$. We can compute this by writing

$$X^{(k)} = \frac{1}{k}(X_1 + X_2 + \ldots X_k)$$

where X_1 is the value of the first ticket drawn from the urn, etc. We then have

$$X^{(k)^2} = \left(\frac{1}{k}\right)^2 \left(\begin{array}{l} {X^2}_1 + {X^2}_2 + \ldots {X^2}_k + X_1 X_2 + \ldots \\ X_1 X_k + X_2 X_1 + \ldots X_2 X_k + \ldots X_{k-1} X_k \end{array} \right).$$

Expectations are linear, so we have that

$$\mathbb{E}\left[X^{(k)}\right]^2 = \left(\frac{1}{k}\right)^2 \left(\begin{array}{l} \mathbb{E}\left[{X^2}_1\right] + \mathbb{E}\left[{X^2}_2\right] + \ldots \mathbb{E}\left[{X^2}_k\right] + \mathbb{E}[X_1X_2] + \\ \ldots \mathbb{E}[X_1X_k] + \mathbb{E}[X_2X_1] + \ldots \mathbb{E}[X_2X_k] + \ldots \mathbb{E}[X_{k-1}X_k] \end{array} \right).$$

The *order* in which the tickets are drawn from the urn doesn't matter, because each time we draw a ticket we draw from the same urn. This means that $\mathbb{E}\left[{X^2}_1\right] = \mathbb{E}\left[{X^2}_2\right] = \ldots \mathbb{E}\left[{X^2}_k\right]$. You can think of this term as the expected value of the random variable generated by: drawing a single number out of the urn; squaring that number; and reporting the square. Notice that $\mathbb{E}\left[{X^2}_1\right] = \mathbb{E}\left[(X^{(1)})^2\right]$ (look at the definition of $X^{(1)}$).

Because the *order* doesn't matter, we also have that $\mathbb{E}[X_1X_2] = \mathbb{E}[X_1X_3] = \ldots \mathbb{E}[X_{k-1}X_k]$. You can think of this term as the expected value of the random variable generated by: drawing a number out of the urn; writing it down; returning it to the urn; then drawing a second number from the urn; and reporting the product of these two numbers. So we can write

$$\mathbb{E}\left[{X^{(k)}}^2\right] = (\frac{1}{k})^2 \left(k\mathbb{E}\left[(X^{(1)})^2\right] + k(k-1)\mathbb{E}[X_1X_2]\right)$$

and these two terms are quite easy to evaluate.

Worked example 9.14 *Urn variances*

Show that

$$\mathbb{E}\left[(X^{(1)})^2\right] = \frac{\sum_{i=1}^{N_p} x_i^2}{N_p} = \mathsf{popsd}\left(\{x\}\right)^2 + \mathsf{popmean}\left(\{x\}\right)^2$$

Solution: First, we have $(X^{(1)})^2$ is the number obtained by taking a ticket out of the urn and squaring its data item. Now

$$\mathsf{popsd}\left(\{x\}\right)^2 = \mathbb{E}\left[(X^{(1)})^2\right] - \mathbb{E}\left[X^{(1)}\right]^2 = \mathbb{E}\left[(X^{(1)})^2\right] - \mathsf{popmean}\left(\{x\}\right)^2$$

so

$$\mathbb{E}\left[(X^{(1)})^2\right] = \mathsf{popsd}\left(\{x\}\right)^2 + \mathsf{popmean}\left(\{x\}\right)^2$$

Worked example 9.15 *Urn variances*

Show that

$$\mathbb{E}[X_1 X_2] = \mathsf{popmean}\left(\{x\}\right)^2$$

Solution: This looks hard, but isn't. Recall from the facts in chapter 6 (useful facts 2, page 134) that if X and Y are independent random variables, $\mathbb{E}[XY] = \mathbb{E}[X]\mathbb{E}[Y]$. But X_1 and X_2 are independent — they are different random draws from the same urn. So

$$\mathbb{E}[X_1 X_2] = \mathbb{E}[X_1]\mathbb{E}[X_2]$$

but $\mathbb{E}[X_1] = \mathbb{E}[X_2]$ (they are draws from the same urn) and $\mathbb{E}[X] = \mathsf{popmean}\left(\{x\}\right)$. So

$$\mathbb{E}[X_1 X_2] = \mathsf{popmean}\left(\{x\}\right)^2.$$

Now

$$\begin{aligned}
\mathbb{E}\left[(X^{(k)})^2\right] &= \frac{k\mathbb{E}\left[(X^{(1)})^2\right] + k(k-1)\mathbb{E}[X_1 X_2]}{k^2} \\
&= \frac{\mathbb{E}\left[(X^{(1)})^2\right] + (k-1)\mathbb{E}[X_1 X_2]}{k} \\
&= \frac{\left(\mathsf{popsd}\left(\{x\}\right)^2 + \mathsf{popmean}\left(\{x\}\right)^2\right) + (k-1)\mathsf{popmean}\left(\{x\}\right)^2}{k} \\
&= \frac{\mathsf{popsd}\left(\{x\}\right)^2}{k} + \mathsf{popmean}\left(\{x\}\right)^2
\end{aligned}$$

so we have

$$\begin{aligned}
\mathsf{var}\left[X^{(k)}\right] &= \mathbb{E}\left[(X^{(k)})^2\right] - \mathbb{E}\left[X^{(k)}\right]^2 \\
&= \frac{\mathsf{popsd}\left(\{x\}\right)^2}{k} + \mathsf{popmean}\left(\{x\}\right)^2 - \mathsf{popmean}\left(\{x\}\right)^2 \\
&= \frac{\mathsf{popsd}\left(\{x\}\right)^2}{k}.
\end{aligned}$$

This is a very useful result which is well worth remembering together with our facts on the sample mean, so we'll put them in a box together.

Useful Fact: 9.5 *The sample mean*

The sample mean is a random variable. Write $X^{(k)}$ for the mean of k samples. We have that:

$$\begin{aligned}
\mathbb{E}\left[X^{(k)}\right] &= \mathsf{popmean}\left(\{x\}\right) \\
\mathsf{var}\left[X^{(k)}\right] &= \frac{\mathsf{popsd}\left(\{x\}\right)^2}{k} \\
\mathsf{std}\left(X^{(k)}\right) &= \frac{\mathsf{popsd}\left(\{x\}\right)}{\sqrt{k}}
\end{aligned}$$

The consequence is this: If you draw k samples, the standard deviation of your estimate of the mean is

$$\frac{\mathsf{popsd}\left(\{x\}\right)}{\sqrt{k}}$$

which means that (a) the more samples you draw, the better your estimate becomes and (b) the estimate improves rather slowly — for example, to halve the standard deviation in your estimate, you need to draw four times as many samples. The standard deviation of the estimate of the mean is often known as the **standard error** of the mean. This allows us to draw a helpful distinction: the population has a standard deviation, and our estimate of its mean (or other things — but we won't go into this) has a standard error.

Notice we cannot state the standard error of our estimate exactly, because we do not know $\mathsf{popsd}\left(\{x\}\right)$. But we could make a good estimate of $\mathsf{popsd}\left(\{x\}\right)$, by computing the standard deviation of the examples that we have. It is now helpful to have some notation for the particular sample we have. I will write $\sum_{i \in \text{sample}}$ for a sum over the sample items, and we will use

$$\mathsf{mean}\left(\{x\}\right) = \frac{\sum_{i \in \text{sample}} x_i}{k}$$

for the mean of the sample — that is, the mean of the data we actually see; this is consistent with our old notation, but there's a little reindexing to keep track of the fact we don't see all of the population. Similarly, I will write

$$\mathsf{std}\left(x\right) = \sqrt{\frac{\sum_{i \in \text{sample}} (x_i - \mathsf{mean}\left(\{x_i\}\right))^2}{k}}$$

for the sample standard deviation. Again, this is the standard deviation of the data we actually see; and again, this is consistent with our old notation, again with a little reindexing to keep track of the fact we don't see all of the population. We

could estimate

$$\mathsf{popsd}\left(\{x\}\right) \approx \mathsf{std}\left(x\right)$$

and as long as we have enough examples, this estimate is good. If the number of samples k is small, it is better to use

$$\mathsf{popsd}\left(\{x\}\right) \approx \sqrt{\frac{\sum_{i \in \text{sample}} (x_i - \mathsf{mean}\left(\{x\}\right))^2}{k-1}}.$$

In fact, much more is known about the distribution of $X^{(k)}$.

9.3.3 The Probability Distribution of the Sample Mean

The sample mean is a random variable. We know an expression for its mean, and we can estimate its variance. In fact, we can determine its probability distribution, though I won't do this rigorously. In section 7.4.3, I mentioned that adding a number of independent random variables almost always got you a normal random variable, a fact sometimes known as the central limit theorem. I didn't prove it, and I'm not going to now. But when we form $X^{(k)}$, we're adding random variables. This means that $X^{(k)}$ is a normal random variable, for sufficiently big k (for some reason, $k > 30$ is usually seen as right).

This is important, because it has the following consequence. Draw a large number of different samples of k elements from the population. Each is a dataset of k items. Compute $\mathsf{mean}\left(\{x\}\right)$ for each, and regard the resulting numbers $e_1, \ldots, e_r$ as data items. Convert the e_i to standard coordinates s_i, where

$$s_i = \frac{(e_i - \mathsf{mean}\left(\{e_i\}\right))}{\mathsf{std}\left(e_i\right)}$$

(i.e. by subtracting the mean of the e_i, and dividing by their standard deviation). Now construct a construct a histogram of the s. If r is sufficiently large, the histogram will be close to the standard normal curve.

9.3.4 When The Urn Model Works

In our model, there was a population of N_p data items x_i, and we saw k of them, chosen at random. In particular, each choice was fair (in the sense that each data item had the same probability of being chosen) and independent. These assumptions are very important for our analysis to apply. If our data does not have these properties, bad things can happen.

For example, assume we wish to estimate the percentage of the population that has beards. This is a mean (the data items take the value 1 for a person with a beard, and 0 without a beard). If we select people according to our model, then ask them whether they have a beard, then our estimate of the percentage of beards should behave as above.

The first thing that should strike you is that it isn't at all easy to select people according to this model. For example, we might select phone numbers at random, then call and ask the first person to answer the phone whether they have a beard; but many children won't answer the phone because they are too small. The next

important problem is that errors in selecting people can lead to massive errors in your estimate. For example, imagine you decide to survey all of the people at a kindergarten on a particular day; or all of the people in a women's clothing store; or everyone attending a beard growing competition (they do exist). In each case, you will get an answer that is a very poor estimate of the right answer, and the standard error might look very small. Of course, it is easy to tell that these cases are a bad choice.

It may not be easy to tell what a good choice is. You should notice the similarity between estimating the percentage of the population that wears a beard, and estimating the percentage that will vote for a particular candidate. There is a famous example of a survey that mispredicted the result of the Dewey-Truman presidential election in 1948; poll-takers phoned random phone numbers, and asked for an opinion. But at that time, telephones tended to be owned by a small percentage of rather comfortable households, who tended to prefer one candidate, and so the polls mispredicted the result rather badly.

Sometimes, we don't really have a choice of samples. For example, we might be presented with a small dataset of (say) human body temperatures. If we can be satisfied that the people were selected rather randomly, we might be able to use this dataset to predict expected body temperature. But if we knew that the subjects had their temperatures measured because they presented themselves at the doctor with a suspected fever, then we most likely cannot use it to predict expected body temperature.

One important and valuable case where this model works is in simulation. If you can guarantee that your simulations are independent (which isn't always easy), this model applies to estimates obtained from a simulation. Notice that it is usually straightforward to build a simulation so that the i'th simulation reports an x_i where $\mathsf{popmean}(\{x\})$ gives you the thing you want to measure. For example, imagine you wish to measure the probability of winning a game; then the simulation should report one when the game is won, and zero when it is lost. As another example, imagine you wish to measure the expected number of turns before a game is won; then your simulation should report the number of turns elapsed before the game was won.

9.4 WHAT YOU SHOULD REMEMBER - NEED SOMETHING

PROBLEMS

Maximum Likelihood Methods

9.1. Fitting a Normal Distribution You are given a dataset of N numbers. Write x_i for the i'th number. You wish to model this dataset with a normal distribution.

- **(a)** Show the maximum likelihood estimate of the mean of this distribution is $\mathsf{mean}(\{x\})$.
- **(b)** Show the maximum likelihood estimate of the standard deviation of this distribution is $\mathsf{std}(x)$.
- **(c)** Now assume that all of these numbers take the same value - what happens to your estimate of the standard deviation?

9.2. Fitting a Poisson Distribution You count the number of times that the annoying "MacSweeper" popup window appears per hour when you surf the web. You wish to model these counts with a Poisson distribution. On day 1, you surf for 4 hours, and see counts of $3, 1, 4, 2$ (in hours 1 through 4 respectively). On day 2, you surf for 3 hours, and observe counts of $2, 1, 2$. On day 3, you surf for 5 hours, and observe counts of $3, 2, 2, 1, 4$. On day 4, you surf for 6 hours, but keep only the count for all six hours, which is 13. You wish to model the intensity in counts per hour.

(a) What is the maximum likelihood estimate of the intensity for each of days 1, 2, and 3 *separately*?

(b) What is the maximum likelihood estimate of the intensity for day 4?

(c) What is the maximum likelihood estimate of the intensity for all days taken together?

9.3. Fitting a Geometric Model You wish to determine the number of zeros on a roulette wheel without looking at the wheel. You will do so with a geometric model. Recall that when a ball on a roulette wheel falls into a non-zero slot, odd/even bets are paid; when it falls into a zero slot, they are not paid. There are 36 non-zero slots on the wheel.

(a) Assume you observe a total of r odd/even bets being paid before you see a bet not being paid. What is the maximum likelihood estimate of the number of slots on the wheel?

(b) How reliable is this estimate? Why?

(c) You decide to watch the wheel k times to make an estimate. In the first experiment, you see r_1 odd/even bets being paid before you see a bet not being paid; in the second, r_2; and in the third, r_3. What is the maximum likelihood estimate of the number of slots on the wheel?

9.4. Fitting a Binomial Model You encounter a deck of Martian playing cards. There are 87 cards in the deck. You cannot read Martian, and so the meaning of the cards is mysterious. However, you notice that some cards are blue, and others are yellow.

(a) You shuffle the deck, and draw one card. It is yellow. What is the maximum likelihood estimate of the fraction of blue cards in the deck?

(b) You repeat the previous exercise 10 times, replacing the card you drew each time before shuffling. You see 7 yellow and 3 blue cards in the deck. What is the maximum likelihood estimate of the fraction of blue cards in the deck?

Bayesian Methods

9.5. Zeros on a Roulette Wheel We now wish to make a more sophisticated estimate of the number of zeros on a roulette wheel without looking at the wheel. We will do so with Bayesian inference. Recall that when a ball on a roulette wheel falls into a non-zero slot, odd/even bets are paid; when it falls into a zero slot, they are not paid. There are 36 non-zero slots on the wheel. We assume that there number of zeros is one of $\{0, 1, 2, 3\}$. We assume that these cases have prior probability $\{0.1, 0.2, 0.4, 0.3\}$.

(a) Write n for the event that, in a single spin of the wheel, an odd/even bet will not be paid (equivalently, the ball lands in one of the zeros). Write z for the number of zeros in the wheel. What is $P(n|z)$ for each of the possible values of z (i.e. each of $\{0, 1, 2, 3\}$)?

(b) Under what circumstances is $P(z = 0|\text{observations})$ NOT 0?

(c) You observe 36 independent spins of the same wheel. A zero comes up in 2 of these spins. What is $P(z|\text{observations})$?

9.6. A Normal Distribution You are given a dataset of 3 numbers, $-1, 0, 20$. You wish to model this dataset with a normal distribution with unknown mean μ and standard deviation 1. You will make an MAP estimate of μ. The prior on μ is normal, with mean 0 and standard deviation 10.

(a) What is the MAP estimate of μ?

(b) A new datapoint, with value 1, arrives. What is the new MAP estimate of μ?

Samples and Populations

9.7. The Average Mouse You wish to estimate the average weight of a mouse. You obtain 10 mice, sampled uniformly at random and with replacement from the mouse population. Their weights are $21, 23, 27, 19, 17, 18, 20, 15, 17, 22$ grams respectively.

(a) What is the best estimate for the average weight of a mouse, from this data?

(b) What is the standard error of this estimate?

(c) How many mice would you need to reduce the standard error to 0.1?

9.8. Sample Variance and Standard Error You encounter a deck of Martian playing cards. There are 87 cards in the deck. You cannot read Martian, and so the meaning of the cards is mysterious. However, you notice that some cards are blue, and others are yellow.

(a) You shuffle the deck, and draw one card. You repeat this exercise 10 times, replacing the card you drew each time before shuffling. You see 7 yellow and 3 blue cards in the deck. As you know, the maximum likelihood estimate of the fraction of blue cards in the deck is 0.3. What is the standard error of this estimate?

(b) How many times would you need to repeat the exercise to reduce the standard error to 0.05?

CHAPTER 10

Inference: Intervals and Testing

Although a point estimate is the best estimate available, it could still be wrong. This doesn't always matter. But in some applications, it can be important to know what range a parameter could take, and still be consistent with the data. We have seen this possibility before, in example 8, where I graphed the posterior — you could look at that graph and see that p could be in a fairly large range of values.

Being able to estimate an interval of values is important, particularly when there are safety or legal considerations to worry about. Imagine you have a machine that fills cereal boxes. Each box gets a quantity of cereal that is random, but has low variance. If the weight of cereal in any box is below the amount printed on the label, you might be in trouble. When you choose the amount to print, a point estimate of the mean weight of cereal might not be particularly helpful. If that estimate is a little low, you could have problems. Instead, what you'd like to know is an interval that the mean lies in with very high probability, so you can print a number that is smaller than the smallest in the interval.

Such intervals are usually known as **confidence intervals**. There are a variety of techniques for estimating them, depending on the particular problem. In section 10.1, I describe straightforward methods to construct confidence intervals. Confidence intervals for sample means are easily constructed using our understanding of standard error. Similarly, the posterior yields confidence intervals in a straightforward way when we use Bayesian inference. In other cases we might not have a good estimate of standard error; but we can use simulation to construct one (section 10.2). That section also shows how to exploit simulation methods to estimate confidence intervals for maximum likelihood parameter estimates.

Now imagine we hypothesize that (say) a population has a given mean. We can then ask how large a confidence interval we have to draw around the sample mean to include the hypothesis. If that interval is very large, then we would have to have drawn a very unusual sample to see the sample mean that we see. This means that our data does not support the hypothesis. This is an extremely fruitful line of reasoning, because it allows us to evaluate evidence in support of a model. Section 10.3 describes methods to do so. In fact, we can evaluate evidence in support of a *model*, too, and section 10.4 shows how to do this.

10.1 STRAIGHTFORWARD CONSTRUCTIONS OF CONFIDENCE INTERVALS

A **statistic** is a function of a dataset. One example of a statistic is the mean of a sample. Yet another is the MAP estimate of a parameter (which you get by computing some numbers from the dataset alone). We observe the value of a statistic, which we can compute from our dataset. But the dataset is random, because it is either a sample from a population or an IID sample from a distribution mode. This means that we should think of the statistic as the observed value of a

random variable — if we had a different sample from the same population (resp. IID sample from the same distribution) we would compute a different value. In these two cases, we know the observed value of the statistic *and* a model for the probability distribution of the statistic, as a random variable. This means we can construct confidence intervals relatively easily.

10.1.1 Confidence Intervals for Population Means

Assume we have a population, and we wish to know a confidence interval for its mean. We draw a sample $\{x\}$ of k items, and compute the sample mean. This is the value of a random variable — random, because it depends on the randomly drawn sample — whose probability distribution we know. This knowledge is very powerful, because it tells us how close to the true mean our estimate is likely to be. The reasoning looks like this. Our estimate of the unknown number $\mathsf{popmean}(\{x\})$ is the mean of the sample we have, which we write $\mathsf{mean}(\{x\})$. We know that the error — which is

$$\mathsf{mean}(\{x\}) - \mathsf{popmean}(\{x\})$$

— is a normal random variable, with mean 0 and standard deviation $\frac{\mathsf{popsd}(\{x\})}{\sqrt{k}}$. We estimate this standard deviation as $\frac{\mathsf{std}(x)}{\sqrt{k}}$. We can scale the error by the standard deviation to get

$$\hat{M} = \frac{\mathsf{mean}(\{x\}) - \mathsf{popmean}(\{x\})}{\left(\frac{\mathsf{popsd}(\{x\})}{\sqrt{k}}\right)} \approx \frac{\mathsf{mean}(\{x\}) - \mathsf{popmean}(\{x\})}{\left(\frac{\mathsf{std}(x)}{\sqrt{k}}\right)}$$

which is a standard normal random variable. But we know rather a lot about the behaviour of standard normal random variables. In particular, we have that:

- About 68% of the time, the value of a standard normal random variable is within one standard deviation of the mean.
- About 95% of the time, the value of a standard normal random variable is within two standard deviations of the mean.
- About 99% of the time, the value of a standard normal random variable is within three standard deviations of the mean.

In turn, this means that

- About 68% of the time, $\mathsf{abs}(\mathsf{mean}(\{x\}) - \mathsf{popmean}(\{x\})) \leq \frac{\mathsf{popsd}(\{x\})}{k}$.
- About 95% of the time, $\mathsf{abs}(\mathsf{mean}(\{x\}) - \mathsf{popmean}(\{x\})) \leq 2\frac{\mathsf{popsd}(\{x\})}{k}$.
- About 99% of the time, $\mathsf{abs}(\mathsf{mean}(\{x\}) - \mathsf{popmean}(\{x\})) \leq 3\frac{\mathsf{popsd}(\{x\})}{k}$.

This means that we can construct a **confidence interval** for our estimate. For about 68% of samples, the sample mean will lie in the interval between $\mathsf{mean}(\{x\}) - \frac{\mathsf{std}(x)}{k}$ and $\mathsf{mean}(\{x\}) + \frac{\mathsf{std}(x)}{k}$, and so on. Notice that we have just accepted that the approximation $\mathsf{std}(x) \approx \mathsf{popsd}(\{x\})$ is harmless, which it is. We can plot the confidence interval by drawing **error bars** — draw a bar one (or two, or three)

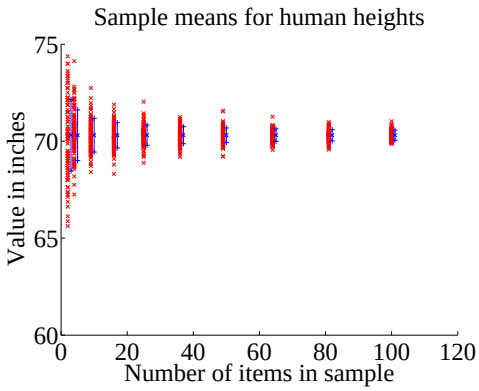

FIGURE 10.1: *I took the heights dataset. I then formed sampled elements with replacement to form random subsets of sizes* $(2, 4, 9, 16, \ldots, 100)$. *For each of 100 subsets of each size, I computed the sample mean — these are shown as x's on the plot. I then computed the population mean, and the standard error as measured by the* population *standard deviation. The x to the side of each column is the population mean, and the vertical bars are one standard error above and below the population mean. Notice how (a) the sample means vary less as the sample gets bigger and (b) the sample means largely lie within the error bars.*

standard deviations up and down from the estimate. We interpret this interval as representing the effect of sampling uncertainty on our estimate. If the urn model really did apply, then the confidence intervals have the property that the true mean lies inside the interval for about 68% of possible samples (for one standard error error bars; or 95% for two; etc.).

It is quite straightforward, and informative, to verify these ideas with a simulation. I used the heights column from the bodyfat dataset (from `http://www2.stetson.edu/~jrasp/data.htm`; look for bodyfat.xls). I removed the single height outlier. I simulated the population using the whole dataset (251 items), then drew numerous samples of various sizes, with replacement. I computed the mean of each of these sets of samples. Figure 10.1 shows a scatter plot of sample means for different samples, using a set of sizes $(2, 4, 9, 16, \ldots, 100)$. I have also plotted the population mean, and the true 1-standard error bars (i.e. using the population standard deviation) for each of these sample sizes. Notice how most sample means lie within the 1-standard error bars, as they should.

Figure 10.2 shows a sample mean and the computed standard error bars for a set of different sized samples. I have also shown the population mean. Notice how the population mean is usually within the computed standard error bars. Notice also how the computed standard error bars (i.e. using the sample standard deviation) are a bit small for small samples, but compare well with the true values for large samples. The problem here is that, for small samples, the sample standard deviation can be a fairly poor estimate of the population standard deviation. In the figure, I show a scatter plot of standard error estimated using sample standard deviation against standard error evaluated using population standard deviation.

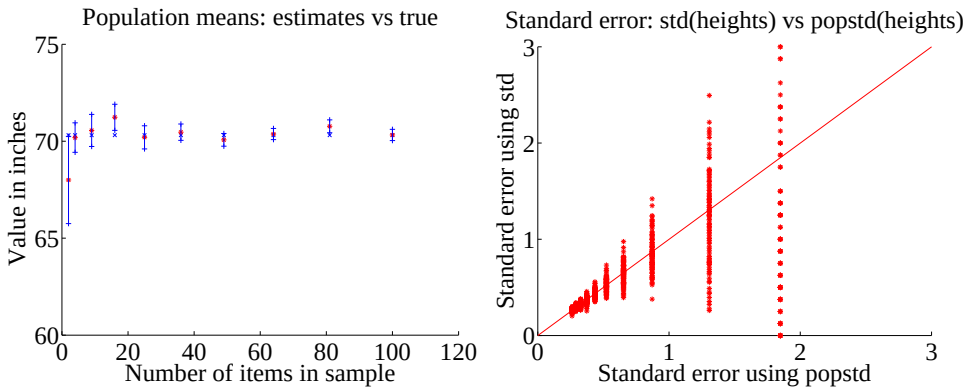

FIGURE 10.2: *On the* **right**, *I chose one sample at random of each size; the sample mean is shown as a *. There are error bars (one standard error above and below) around the sample mean. These error bars are computed from the* sample *standard deviation. The population mean is the x. Notice how the population mean is within the error bars most, but not all, of the time (about 68% of the time, as they should be). The sample mean is rather a good estimate of the population mean, and the standard error is quite a reliable estimate of how well the sample mean represents the population mean when the sample is large enough. When the sample is small, the estimate is poor, as you can see on the* **right**. *This is a scatter plot of the true standard error (using the population standard deviation) against the estimate (using the sample standard deviation). When the sample is small, the standard error is big, and the estimate is poor.*

When the standard error is small (because the sample is large), the two are quite close; when it is large (because the sample is small), they can be quite different. This suggests, correctly, that accurate inference from small samples can be tricky. The right thing to do is get more data — it is optimistic to hope for a good estimate of average human height from two people, after all.

Now we might reasonably ask another question. We should like to know a confidence interval such that the true mean lies within it for p of the samples. This is equivalent to asking what is the u such that

$$\int_{-u}^{u} \frac{1}{\sqrt{2\pi}} \exp\left(-x^2/2\right) dx = p$$

(i.e. what is the range of values about zero such that $p\%$ of standard normal random variables lies in this range). Such numbers can be extracted from the inverse of the error function (which is known as the **inverse error function**). An alternative is to look in statistical tables, which usually give u values for a useful set of p's.

Worked example 10.1 *Speed of light*

Give a 95% confidence interval for the time of a light transit across a fixed distance, based on the dataset at `http://lib.stat.cmu.edu/DASL/Datafiles/SpeedofLight.html`.

Solution: The first thing to notice about this dataset is it seems to contain two negative times; but this is illusory. Look closely by tracking down on the web the paper referred to in the story (Stigler, S.M., "Do robust estimators work with real data?" *Annals of Statistics*, 5 (1977), pp. 1055-1078.). You will find that Newcomb measured the the passage of time for light to cross a fixed distance. He then subtracted a constant, and reported the result. Write t for the true measurement of time, in microseconds, and n for the number in the dataset. We have is

$$t = 24.8 + n \times 1e - 3.$$

Newcomb omitted the smallest value in the dataset, which is very different from all others. **Omitting** the smallest value gives a mean of 24.82729, a standard deviation of 0.0062, and a standard error of 0.00078. So with 95% confidence the value lies between 24.82573 and 24.82885. **Keeping** the smallest value gives a mean of 24.8262, a standard deviation of 0.0107 and a standard error of 0.0013. So with 95% confidence the value lies between 24.8236 and 24.8288.

Notice the temptation to drop an outlier in example 1- it really makes the measurement look better. These days, the speed of light is known precisely. It's used to define the metre, so we would take the position that Newcomb was measuring the distance of the transit.

10.1.2 Bayesian Confidence Intervals

We can use the posterior to evaluate the probability that the true value of a parameter lies within an interval. Assume we want to know for $a < b$, the probability that the parameter θ lies in that interval. In many cases, we can compute

$$\int_a^b p(\theta|\mathcal{D})d\theta,$$

which supplies the relevant probability.

Worked example 10.2 *Flipping a coin - II*

We have a coin with probability θ of coming up heads when flipped. We start knowing nothing about θ except that, being a probability, $0 \leq \theta \leq 1$. We use a uniform prior on θ. We then flip the coin 10 times, and see 7 heads (and 3 tails). What is $P(\{\theta \in [0.5, 0.8]\} | \mathcal{D})$?

Solution: We could answer this question by computing

$$\int_{0.5}^{0.8} p(\theta|\mathcal{D}) d\theta.$$

Recall that a Beta distribution with $\alpha = \beta = 1$ is uniform, and look at example 11. The posterior is

$$P(\theta|10, 7, 1, 1) = \frac{\Gamma(12)}{\Gamma(8)\Gamma(4)} \theta^7 (1-\theta)^3.$$

I evaluated this integral numerically, and got 0.73.

In example 2, I determined that $P(\{\theta \in [0.5, 0.8]\} | \mathcal{D})$ was 0.73 Instead, I might wish to specify $0 \leq u < 0.5$, and then find an interval $[a, b]$ such that $P(\{\theta \in [a, b]\} | \mathcal{D}) = 1 - 2u$. Notice that this interval is not unique. Notice that

$$P(\{\theta \in [a, b]\} | \mathcal{D}) = 1 - P(\{\theta \leq a\} | \mathcal{D}) - P(\{\theta \geq b\} | \mathcal{D})$$

We will choose a such that

$$P(\{\theta \leq a\}) = \int_{-\infty}^{a} P(\theta|\mathcal{D}) d\theta = u$$

and b such that

$$P(\{\theta \geq b\} | \mathcal{D}) = \int_{b}^{\infty} P(\theta|\mathcal{D}) d\theta = u.$$

Actually obtaining values for a and b might be quite difficult. One strategy is to search a range of values.

Worked example 10.3 *Flipping a coin - II*

We have a coin with probability θ of coming up heads when flipped. We start knowing nothing about θ except that, being a probability, $0 \leq \theta \leq 1$. We use a uniform prior on θ. We then flip the coin 10 times, and see 7 heads (and 3 tails). Construct an interval $[a, b]$ such that $P(\{\theta \leq a\}|\mathcal{D}) \approx 0.05$ and $P(\{\theta \geq b\}|\mathcal{D}) \approx 0.05$.

Solution: I wrote a brief program to compute the relevant integrals numerically, then searched using interval halving. I found $P(\{\theta \leq 0.435\}|\mathcal{D}) \approx 0.05$ and $P(\{\theta \geq 0.865\}|\mathcal{D}) \approx 0.05$; this means the interval is $[0.435, 0.865]$.

10.2 USING SIMULATION TO CONSTRUCT INTERVALS

The analysis of some problems is difficult. But the idea of a confidence interval is naturally associated with repeated experiments. We can use simulations to estimate confidence intervals when analysis is difficult. Generally, we see the dataset as one of many possible dataset we could have seen. We must then assess what our estimate would be like for the other possible datasets. But what are those possible datasets to be?

When the data is explained by a parametric probability model, we can use that model to produce other possible datasets. If we compute a maximum likelihood estimate $\hat{\theta}$ of the parameters of the model, we can draw IID samples *from the model*. We then look at the estimate from that new dataset; the spread of the estimates yields our confidence interval.

When we have no applicable probability model of the dataset, we can resample the dataset to simulate the effects of sampling error. This strategy allows us to build confidence intervals for properties like medians.

10.2.1 Constructing Confidence Intervals for Parametric Models

Assume we have a dataset $\mathcal{D} = \{\mathbf{x}\}$, and a probability model we believe applies to that dataset. We can estimate model parameters, we can maximize the likelihood function $L(\theta)$.

A $(1 - 2\alpha)$ confidence interval for a parameter is an interval $[c_\alpha, c_{(1-\alpha)}]$. We construct the interval such that, if we were to perform a very large number of repetitions of our original experiment then estimate a parameter value for each, c_α would be the α quantile and $c_{(1-\alpha)}$ would be the $(1-\alpha)$ quantile of those parameter values. We interpret this to mean that, with confidence $(1 - 2\alpha)$, the correct value of our parameter lies in this interval. This definition isn't really watertight. How do we perform a very large number of repetitions? If we don't, how can we tell how good the confidence interval is? Nonetheless, we can construct intervals that illustrate how sensitive our original inference is to the data that we have.

There are a variety of methods to construct confidence intervals. We will focus on simulation. The algorithm should feel so natural to you that you may already have guessed what to do. First, we compute the maximum likelihood estimate of the parameters, $\hat{\theta}$. We assume this estimate is right, but need to see how our estimates of $\hat{\theta}$ would vary with different collections of data from our model with that parameter value. So we compute a collection of simulated datasets, $\mathcal{D}_i$, each the same size as the original dataset. We obtain these by simulating $P(d|\hat{\theta})$. Next, we compute a maximum likelihood estimate for each of these simulated datasets, $\hat{\theta}_i$. Finally, we compute the relevant percentiles of these datasets. The result is our interval.

Procedure: 10.1 *Estimating Confidence Intervals for Maximum Likelihood Estimates using Simulation*

Assume we have a dataset $\mathcal{D} = \{x\}$ of N items. We have a parametric model of this data $p(x|\theta)$, and write the likelihood $\mathcal{L}(\theta; \mathcal{D}) = P(\mathcal{D}|\theta)$. We construct $(1 - 2\alpha)$ confidence intervals using the following steps.

1. Compute the maximum likelihood estimate of the parameter, $\hat{\theta} = \underset{\theta}{\text{argmax}}\ \mathcal{L}(\theta; \mathcal{D})$.
2. Construct R simulated datasets $\mathcal{D}_i$, each consisting of N IID samples drawn from $p(x|\hat{\theta})$.
3. For each such dataset, compute $\hat{\theta}_i = \underset{\theta}{\text{argmax}}\ \mathcal{L}(\theta; \mathcal{D})$.
4. Obtain $c_\alpha(\hat{\theta}_i)$, the α'th quantile of the collection $\hat{\theta}_i$ and $c_{(1-\alpha)}(\hat{\theta}_i)$, the $1 - \alpha$'th quantile of the collection $\hat{\theta}_i$.

The confidence interval is $[c_\alpha, c_{(1-\alpha)}]$.

Figure 10.3 shows an example. In this case, I worked with simulated data from a normal distribution. In each case, the normal distribution had mean 0, but there are four different standard deviations (1, 4, 7, and 10). I simulated 10 different datasets from each of these distributions, containing $10, 40, 90, \ldots, 810, 1000$ data items. For each, I computed the maximum likelihood estimate of the mean. This isn't zero, *even though the data was drawn from a zero-mean distribution*, because the dataset is finite. I then estimated the confidence intervals for each using 10, 000 simulated datasets of the same size. I show 95% confidence intervals for the cases $\sigma = 1$, $\sigma = 4$, $\sigma = 10$ in the figure, plotted against dataset size. Notice that these aren't symmetric about zero, because the maximum likelihood estimate isn't zero. They shrink as the dataset grows, but slowly. They are bigger when the standard deviation is bigger. It should seem reasonable that you can't expect an accurate estimate of the mean of a normal distribution with large standard deviation using only a few data points. One can demonstrate using statistical theory that would take us rather out of our way that the size of these intervals should behave like

$$\frac{\sigma}{\sqrt{N}}$$

and Figure 10.3 suggests that they do. I plotted $1/\text{size}^2$; you can see this grows linearly as N grows, and falls very fast as σ grows.

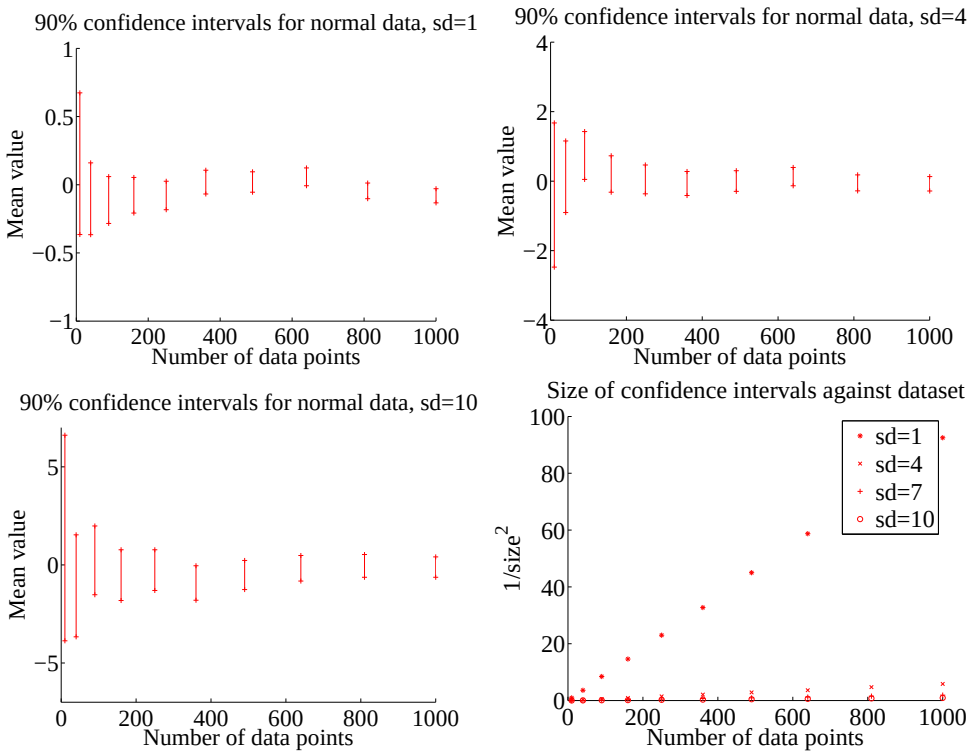

FIGURE 10.3: *Confidence intervals computed for simulated normal data; details in the text.*

Worked example 10.4 *Confidence Intervals by Simulation - II*

Construct a 90% confidence interval for the intensity estimate for the data of example 5 for the cases of 10 observations, 20 observations, and all 30 observations.

Solution: Recall from that example the maximum likelihood estimates of the intensity are 7/10, 22/20, and 29/30 in the three cases. I used the Matlab function `poissrnd` to get 10000 replicates of a dataset of 10 (resp. 20, 30) items from a Poisson distribution with the relevant intensities. I then used `prctile` to get the 5% and 95% percentiles, yielding the intervals

$$\begin{aligned} [0.3, 1.2] &\quad \text{for 10 observations} \\ [0.75, 1.5] &\quad \text{for 20 observations} \\ [0.6667, 1.2667] &\quad \text{for 30 observations} \end{aligned}$$

Notice how having more observations makes the confidence interval smaller.

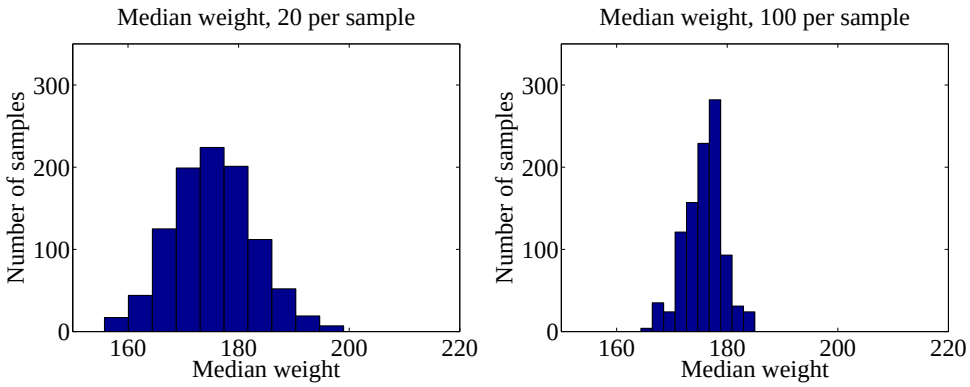

FIGURE 10.4: *I took the weights dataset used all 253 measurements to represent a population. Rather than compute the median of the whole population, I chose to compute the median of a randomly chosen sample. The figures show a histogram of 1000 different values of the median, computed for 1000 different samples (of size 20 on the* **left***, and of size 100 on the* **right***). Notice that (a) there is a moderate amount of variation in the median of the sample; (b) these histograms look normal, and appear to have about the same median; (c) increasing the size of the sample has reduced the spread of the histogram.*

10.2.2 Estimating Standard Error

We were able to produce convenient and useful estimates of standard error for sample means. But what happens if we want to, say, reason about the median of a population? It turns out that estimating standard error becomes difficult mathematically. This is an important problem, because if we had standard error estimates, we could use our methods for building confidence intervals and for testing hypotheses. It turns out that quite simple simulation methods give very good estimates.

For Figure 10.4, I assumed that all 253 weight measurements represented the entire population, then simulated what would happen for different random samples (with replacement) of different sizes. Figure 10.4 suggests that the sample median behaves quite like the sample mean as the random sample changes. Different samples have different medians, but the distribution of values looks fairly normal. When there are more elements in the sample, the standard deviation of median values is smaller. But we have no method to compute this standard deviation. The method I used here doesn't apply in practice, because we don't usually have the whole population.

There is a method, known as the **bootstrap**, which gives a very good estimate of the standard error of any statistic. Assume we wish to estimate the standard error of a statistic $S(\{\mathbf{x}\})$, which is a function of our dataset $\{\mathbf{x}\}$ of N data items. We compute r **bootstrap replicates** of this sample. Each replicate is obtained by sampling the dataset uniformly, and with replacement. One helpful way to think of this is that we are modelling our dataset as a sample of a probability distribution. This distribution, sometimes known as the **empirical distribution**,

has probability $1/N$ at each of the data items we see, and zero elsewhere. Now to obtain replicates, we simply draw new sets of IID samples from this probability distribution.

Notice that the bootstrap replicates are *not* a random permutation of the dataset; instead, we select one data item fairly and at random from the whole dataset N times. This means we expect a particular bootstrap replicate will have multiple copies of some data items, and no copies of others.

We write $\{\mathbf{x}\}_i$ for the i'th bootstrap replicate of the dataset. We now compute

$$\overline{S} = \frac{\sum_i S(\{\mathbf{x}\}_i)}{r}$$

and the standard error estimate for S is given by:

$$\mathsf{stderr}\left(\{S\}\right) = \sqrt{\frac{\sum_i \left[S(\{\mathbf{x}\}_i) - \overline{S}\right]}{r-1}}$$

Procedure: 10.2 *The bootstrap*

Estimate the standard error for a statistic S evaluated on a dataset of N items $\{\mathbf{x}\}$.

1. Compute r bootstrap replicates of the dataset. Write the i'th replicate $\{\mathbf{x}\}_i$ Obtain each by:
   - **(a)** Building a uniform probability distribution on the numbers $1 \ldots N$.
   - **(b)** Drawing N independent samples from this distribution. Write $s(i)$ for the i'th such sample.
   - **(c)** Building a new dataset $\left\{x_{s(1)}, \ldots, x_{s(N)}\right\}$.
2. For each replicate, compute $\sum_i S(\{\mathbf{x}\}_i)$.
3. Compute
$$\overline{S} = \frac{\sum_i S(\{\mathbf{x}\}_i)}{r}$$
4. The standard error estimate for S is given by:
$$\mathsf{stderr}\left(\{S\}\right) = \sqrt{\frac{\sum_i \left[S(\{\mathbf{x}\}_i) - \overline{S}\right]}{r-1}}$$

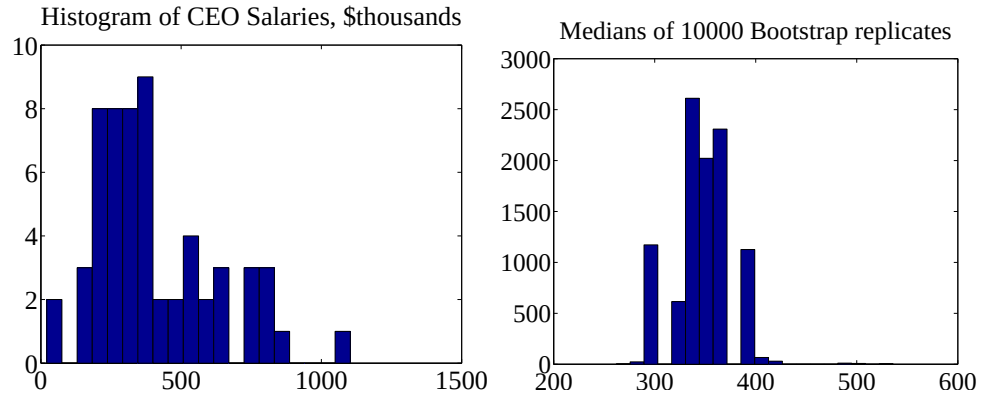

FIGURE 10.5: *On the* **left**, *a histogram of salaries for CEO's of small companies in 1993, from the dataset of* `http://lib.stat.cmu.edu/DASL/Datafiles/ceodat.html`. *On the* **right**, *a histogram of the medians of 10, 000 bootstrap replicates of this data. This simulates the effect of sampling variation on the median; see worked example 5.*

Worked example 10.5 *The bootstrap standard error of the median*

You can find a dataset giving the salaries of CEO's at small firms in 1993 at `http://lib.stat.cmu.edu/DASL/Datafiles/ceodat.html`. Construct a 90% confidence interval for the median salary.

Solution: Salaries are in thousands of dollars, and one salary isn't given (we omit this value in what follows). Figure **??** shows a histogram of the salaries; notice there are some values that look like outliers. This justifies using a median. The median of the dataset is 350 (i.e. $ 350, 000 — this is 1993 data!). I constructed 10, 000 bootstrap replicates. Figure 10.5 shows a histogram of the medians of the replicates. I used the matlab `prctile` function to extract the 5% and 95% percentiles of these medians, yielding the interval between 298 and 390. This means that we can expect that, for 90% of samples of CEO salaries for small companies, the median salar will be in the given range.

Worked example 10.6 *Are Brinks collectors different from city collectors? - I*

Using the dataset of `http://lib.stat.cmu.edu/DASL/Datafiles/brinkdat.html`, test whether Brinks coin collectors are different from city coin collectors (read the story, at that URL).

Solution: There are two types of parking meter in this story (which you should have read!). One is close to the city hall, and is always collected by city employees. The other is far, and for some months was collected by Brinks employees and for some months collected by city employees. I ignored the figures for month 10, because there was a missing digit in the number (don't know how or why). For the meters that are far, coins were collected for 22 months by city employees and for 24 months by Brinks employees.
For the far meters, I constructed 10, 000 bootstrap replicates of each case. Figure 10.6 compares the two sets of replicates, showing the histogram of means for city collections vs Brinks collections. Notice how the means of the two histograms are quite separated, compared to their variance. This suggests, quite strongly, that the Brinks collections are different (the difference between the means is about 1.6 standard deviations). I used the nearby meter figures to check whether there was something special about the months on which Brinks employees collected coins. Again, I constructed 10, 000 bootstrap replicates of each case (nearby coins in a Brinks month vs. nearby coins in a city month). This is plotted in figure 10.6. There's nothing special about the months, from the figure.

10.3 USING THE STANDARD ERROR TO EVALUATE HYPOTHESES

Assume we have a **hypothesis** to work with. For example, we believe that women and men have the same body temperature. We should like to assess the extent to which the evidence we have contradicts the hypothesis. At first glance, this may seem strange to you — surely one wants to assess the extent to which the evidence *supports* the hypothesis — but in fact it's natural. You can't prove that a scientific hypothesis is true; you can only fail to show that it's false. Just one piece of evidence can destroy a scientific hypothesis, but no amount of evidence can remove all doubt.

In our example, we measure the temperatures of a sample of women and of a sample of men. We cannot expect that the sample means are the same, because these are affected by the choice of sample. However, we have a good model of these sample means as random variables; in particular, we know the standard error. We can use this to assess the extent to which the difference in temperatures can be explained solely by the choice of sample. If it is easy to explain the difference in temperatures by choice of sample, then the evidence does not contradict the hypothesis; alternatively, if the difference in temperatures is hard to explain by

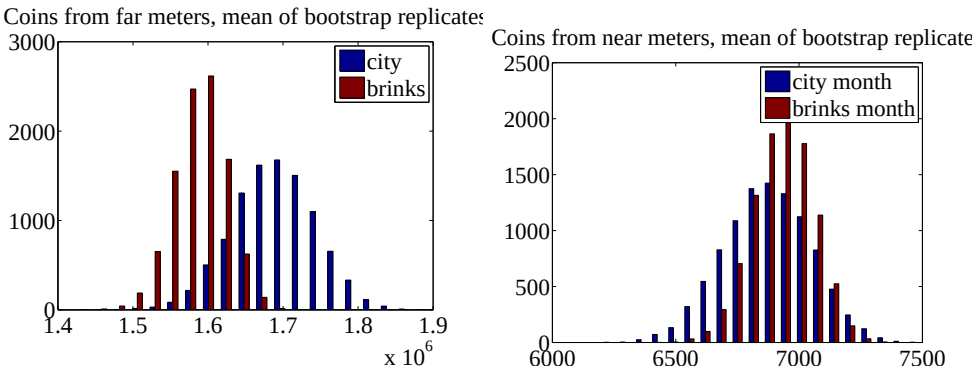

FIGURE 10.6: *On the* **left**, *two histograms comparing the means of bootstrap replicates of the coins collected by Brinks employees and those collected by city employees. Notice how the two histograms are quite separated, compared to their width. This is fairly strong evidence that the two cases are different. The difference is not explained by the months in which the coins are collected. On the* **right**, *two histograms comparing the means of bootstrap replicates of coins collected from nearby meters. These coins were collected by city employees, but I have separated Brinks months from city months. There is no evidence the months are different.*

choice of sample, the evidence contradicts the hypothesis.

There is an important, quite general, line of reasoning here. It is a bad idea to try and explain data using a hypothesis that makes the data you observed a rare event.

Example: 10.1 *Patriot missiles*

I got this example from "Dueling idiots", a nice book by P.J. Nahin, Princeton University Press. Apparently in 1992, the Boston Globe of Jan 24 reported on this controversy. The pentagon claimed that the patriot missile successfully engaged SCUD missiles in 80% of encounters. An MIT physicist, Theodore Postol, pointed out there was a problem. He viewed tapes of 14 patriot/SCUD encounters, with one hit and 13 misses. We can reasonably assume each encounter is independent. We can extract the probability of getting one hit and 13 misses if $P(\text{hit}) = 0.8$ from the binomial model, to get a number around 1e-8. Now you could look at this information and make several arguments: (a) the pentagon is right and the probability really is 0.8, but Postol looked at a really unlucky set of videotapes; (b) the probability is not 0.8, because you would need to fire 14 patriots at 14 SCUD missiles about 1e8 times to see this set of videotapes once; (c) for some reason, the videotapes are not independent — perhaps only unsuccessful encounters get filmed. If Postol viewed tapes at random (i.e. he didn't select only unsuccessful tapes, etc.), then argument (a) is easily dismissed, because the pentagon would have had to be unreasonably unlucky — it's a bad idea to try to explain data with a hypothesis that makes the data very unlikely.

Example: 10.2 *MTG and Shuffling*

You build a deck of 24 lands and 36 spells. You shuffle this deck, and draw a hand of seven cards. You get no lands. You repeat the experiment, and still get no lands. On a third try, you still get no lands. By example 17, this event (three shuffles and draws with no lands in each hand) has probability about 8e-6. You could draw several conclusions: (a) you got unlucky; (b) you miscounted lands somehow; (c) you're shuffling the cards badly (i.e. the cards stick together, or you are bad at shuffling). Conclusion (a) is unattractive, because you would need to have been very unlucky (you'd see this outcome once in roughly 1e5 experiments, where each consists of shuffling and drawing three hands). It's a bad idea to try to explain data with a hypothesis that makes the data very unlikely, so conclusions (b) and (c) have become more plausible.

10.3.1 Does this Population have this Mean?

Imagine we hypothesize that the average human body temperature is 95^o. Write $\overline{T}$ for the random variable evaluated by: collecting a random sample of people, measuring their temperatures, and computing the average of these temperatures. The mean of this random variable is the population mean, and the standard deviation of this random variable is the standard error, which we write s. Now consider the random variable

$$G = \frac{(\overline{T} - 95^o)}{s}.$$

This is a standard normal random variable, if our hypothesis is true.

In practice, we have already chosen a single sample. We can compute the mean temperature of this sample, which we write $\overline{t}$. This represents an observed value of $\overline{T}$. If you're puzzled by the relationship between the two, use an analogy with rolling a die. The random variable is what you could get if you rolled a die; the value is the face that came up when you did roll it. Similarly, $\overline{T}$ is what you could get by choosing a sample, and $\overline{t}$ is the value you did get with the sample you have. In turn, we obtain a value for G, which is

$$g = \frac{(\overline{t} - 95^o)}{s}.$$

Now G is a standard normal random variable, if our hypothesis is true. We can interpret this to mean that for 68% of possible samples, g will take a value between -1 and 1 (etc.), if our hypothesis is true. Equivalently G will take a value between g and $-g$, for a fraction f of all possible samples of the population, where

$$f = \int_{-g}^{g} \exp\left(\frac{-u^2}{2}\right) du.$$

Now assume we see a very large (or very small) value of g. The value of f will be close to one, which implies that most samples from the population will have a g value closer to zero if the hypothesis were true. Equivalently, this says that, if the hypothesis were true, our sample is highly unusual, which implies the data fails to support the hypothesis.

It is easy to think about $p = 1 - f$ than about f. You should think of p as representing the fraction of samples that would give a larger absolute value of g, *if* the null hypothesis was true. If this fraction is very small, then there is significant evidence against the null hypothesis. We have that p — the fraction of experiments that would give us a larger absolute value of g than the one we saw — is given by

$$p = (1 - f) = (1 - \int_{-g}^{g} \exp\left(\frac{-u^2}{2}\right) du) = P(\{G > |\hat{g}|\}) \cup P(\{G < -|\hat{g}|\})$$

and we can compute this number easily, because we know the distribution of G is normal (or nearly so). The fraction is sometimes referred to as a **p-value**.

Once we have the p-value, testing is straightforward. A small value of the fraction means that the outcome we see would be rare *if* the null hypothesis is true. The fraction that we compute should be interpreted an assessment of the

significance of the evidence against the null hypothesis. The p-value is smaller when the evidence against the null hypothesis is stronger; we get to decide how small a p-value means we should reject the null hypothesis.

It is conventional to reject the null hypothesis when the p-value is less than 0.05. This is sometimes called "a significance level of 5%". You should interpret a p-value of 0.05 as meaning that you would see evidence this unusual in about one experiment in twenty if the null hypothesis was true. Sometimes, the p-value is even smaller, and this can be interpreted as very strong evidence the null hypothesis is wrong. A p-value of less than 0.01 allows one to reject the null hypothesis at "a significance level of 1%". Similarly, you should interpret a p-value of 0.01 as meaning that you would see evidence this unusual in about one experiment in a hundred if the null hypothesis was true. We now have a general recipe for testing whether a population has a particular mean, which belongs in a box:

Procedure: 10.3 *Testing whether a population has a given mean, based on a sample*

The initial hypothesis is that the population has a known mean, which we write μ. Write $\{x\}$ for the sample, and k for the sample size.

- Compute the sample mean, which we write $\mathsf{mean}\,(\{x\})$.
- Estimate the standard error s_e using

$$s_e = \frac{\mathsf{std}\,(x)}{\sqrt{k}}.$$

- Compute the **test statistic** using

$$s = \frac{(\mu - \mathsf{mean}\,(\{x\}))}{s_e}.$$

- Compute the p-value, using

$$p = (1 - f) = (1 - \int_{-s}^{s} \exp\left(\frac{-u^2}{2}\right) du)$$

  Equivalently, compute the probability that a standard normal random variable X takes a value greater than s or less than $-s$, using the expression

$$P(\{X > |s|\}) \cup P(\{X < -|s|\}).$$

- The p-value summarizes the extent to which the data contradicts the hypothesis. A small p-value implies that, *if* the hypothesis is true, the sample is very unusual. The smaller the p-value, the more strongly the evidence contradicts the hypothesis.

It is common to think that a hypothesis can be rejected only if the p-value is less than 5% (or some number). You should not think this way; the p-value summarizes the extent to which the data contradicts the hypothesis, and your particular application logic affects how you interpret it.

Worked example 10.7 *Average Human Body Weight*

Assess the null hypothesis that the average human body weight is 175 lb, using the height and weight data set of `http://www2.stetson.edu/~jrasp/data.htm` (called bodyfat.xls).

Solution: The dataset contains 252 samples; the average weight is 178.9lb. We know this average is an estimate of the mean of the original large set of all people. This estimate is a normal random variable whose mean is the mean of the population, and whose standard deviation is given by the standard error.
Our sample is large (which usually means over 30 elements) and so we can estimate the standard error as

$$\frac{\text{standard deviation of sample}}{\sqrt{\text{number in sample}}} = \frac{29.4}{15.9} = 1.9,$$

where the units are lb. The test statistic is

$$\frac{\text{sample mean} - \text{hypothesized population mean}}{\text{standard error}} = \frac{178.9 - 175}{1.9} = 2.05.$$

This is the value of a standard normal random variable. We must compute the fraction of outcomes where the test statistic would be larger than 2.05, or smaller than -2.05. This is

$$2 * \frac{1}{\sqrt{2\pi}} \int_{2.05}^{\infty} \exp\left(\frac{-x^2}{2}\right) dx = 0.02$$

so the p-value is 0.02. We can interpret this as quite strong evidence that the average human body weight is not, in fact, 175lb. This p-value says that, if (a) the average human body weight is 175lb and (b) we repeat the experiment (weigh 252 people and average their weights) 50 times, we would see a number as big as the one we see about once.

Worked example 10.8 *Average BMI*

The BMI (body mass index) is a number intended to capture how much a person's weight deviates from the usual for their height. Assess the null hypothesis that the average human BMI is 27, using the height and weight data set of `http://www2.stetson.edu/~jrasp/data.htm`; bodyfat.xls.

Solution: BMI is computed using

$$BMI = \frac{\text{weight in lb}}{(\text{height in in})^2} \times 703.$$

I did excluded two possible outliers in this dataset (entry 39 has weight 363 and height 72.25; entry 42 has weight 205 and height 29.5). I found a mean BMI of 25.3, and a standard deviation of 3.35. There are 250 items in the dataset, so the standard error is 0.21. The test statistic is

$$\frac{\text{sample mean} - \text{hypothesized population mean}}{\text{standard error}} = \frac{25.3 - 27}{2.1} = -8.1$$

so the p-value is the probability that a standard normal random variable would take a value larger than -8.1 or smaller than 8.1. There is no particular reason to care about the difference between this (extremely small, about 1e-16) number and zero. The evidence is very strongly against the hypothesis.
If one keeps the outliers, one gets a mean BMI of 25.9, and a standard deviation of 9.56. There are 252 items in the dataset, so the standard error is 0.60. Now the p-value is 0.08, suggesting that the evidence is against the hypothesis, but not overwhelmingly. Notice the significant effect of just two datapoints. We might need more data to be sure we could reject the hypothesis.

You should notice an important relationship between our test and the material of section 10.1. When we constructed a confidence interval, we knew the distribution that the sample mean would take as we randomly chose different samples from the population. In turn, this meant we could plot (for example) the range of possible values the population mean would take for (say) 95% of possible samples. When we test a hypothesis about the mean, we are asking what kind of confidence interval we would have to draw around the hypothesized mean to encompass the observed sample mean.

10.3.2 Do Two Populations have the same Mean?

We have two samples, and we need to know whether they come from the same, or different, populations. For example, we might observe people using two different interfaces, measure how quickly they perform a task, then ask are their perfor-

mances different? As another example, we might run an application with no other applications running, and test how long it takes to run some standard tasks. Because we don't know what the operating system, cache, etc. are up to, this number behaves a bit like a random variable, so it is worthwhile to do several experiments, yielding one set of samples. We now do this with other applications running as well, yielding another set of samples — is this set different from the first set? For realistic datasets, the answer is always yes, because they're random samples. A better question is could the differences be the result of chance, or are these datasets really samples of two different populations?

We will ask if the population means are different. It turns out we can answer this rather easily, using some tricks about normal random variables. Assume that the samples are large enough (30 seems to be the magic number here). Write $\{x\}$ for the first dataset, which has size k_x, and $\{y\}$ for the second, which has size k_y. These datasets need not be of the same size Each dataset is a random sample, drawn with replacement, from a population. We should like to assess the evidence that these two populations have the same mean. To do so, we need some simple facts about normal random variables.

Useful Facts: 10.1 *Sums and differences of normal random variables*

Let X_1 be a normal random variable with mean μ_1 and standard deviation σ_1. Let X_2 be a normal random variable with mean μ_2 and standard deviation σ_2. Let X_1 and X_2 be independent. Then we have that:

- for any constant $c_1 \neq 0$, $c_1 X_1$ is a normal random variable with mean $c_1\mu_1$ and standard deviation $c_1\sigma_1$;
- for any constant c_2, $X_1 + c_2$ is a normal random variable with mean $\mu_1 + c_2$ and standard deviation σ_1;
- $X_1 + X_2$ is a normal random variable with mean $\mu_1 + \mu_2$ and standard deviation $\sqrt{\sigma_1^2 + \sigma_2^2}$.

I will not prove these facts; we already know the expressions for means and standard deviations from our results on expectations. The only open question is to show that the distributions are normal. This is easy for the first two results. The third requires a bit of integration that isn't worth our trouble; you could do reconstruct the proof from section 1's notes on sums of random variables and some work with tables of integrals.

Write $X^{(k_x)}$ for the random variable obtained by: drawing a random sample with replacement of k_x elements from the first population, then averaging this sample. Write $Y^{(k_y)}$ for the random variable obtained by: drawing a random sample with replacement of k_y elements from the first population, then averaging this sample. From section 9.3, we know that each random variable is normal, and

we know the means and standard deviations of these random variables. In turn, this means that $X^{(k_x)} - Y^{(k_y)}$ is a normal random variable.

Now write D for $X^{(k_x)} - Y^{(k_y)}$. If the two populations have the same mean, then $\mathbb{E}[D] = 0$. Furthermore,

$$\mathsf{std}\,(D) = \sqrt{\mathsf{std}\left(X^{(k_x)}\right)^2 + \mathsf{std}\left(X^{(k_y)}\right)^2}.$$

We know how to estimate $\mathsf{var}\left[X^{(k_x)}\right]$ — it is the standard error of the sample, and we can estimate it as $\mathsf{std}\,(x)/\sqrt{k_x}$. So we have can estimate $\mathsf{std}\,(D)$ as

$$\mathsf{std}\,(D) \approx \sqrt{\left(\frac{\mathsf{std}\,(x)}{\sqrt{k_x}}\right)^2 + \left(\frac{\mathsf{std}\,(y)}{\sqrt{k_y}}\right)^2}.$$

We can now use the same reasoning we used to test the hypothesis that a population had a particular, known mean. We have identified a number we can compute from the two samples. We know how this number would vary under random choices of sample. If the value we observe is too many standard deviations away from the mean, the evidence is against our hypothesis. If we wanted to believe the hypothesis, we would be forced to believe that the sample is extremely strange. I have summarized this reasoning in box 4.

Procedure: 10.4 *Testing whether two populations have a given mean, based on a sample of each*

The initial hypothesis is that the populations have the same, unknown, mean. Write $\{x\}$ for the sample of the first population, $\{y\}$ for the sample of the second population, and k_x, k_y for the sample sizes.

- Compute the sample means for each population, $\mathsf{mean}\,(\{x\})$ and $\mathsf{mean}\,(\{y\})$.
- Estimate the standard error s_e for each population, using

$$s_{ex} = \frac{\mathsf{std}\,(x)}{\sqrt{k}}, \qquad s_{ey} = \frac{\mathsf{std}\,(y)}{\sqrt{k}}.$$

- Compute the standard error for the difference between the means,

$$s_{ed} = \sqrt{s_{ex}^2 + s_{ey}^2}.$$

- Compute the **test statistic** using

$$s = \frac{(\mathsf{mean}\,(\{x\}) - \mathsf{mean}\,(\{y\}))}{s_{ed}}.$$

- Compute the p-value, using

$$p = (1 - f) = (1 - \int_{-s}^{s} \exp\left(\frac{-u^2}{2}\right) du)$$

  Equivalently, compute the probability that a standard normal random variable X takes a value greater than s or less than $-s$, using the expression

$$P(\{X > |s|\}) \cup P(\{X < -|s|\}).$$

- The p-value summarizes the extent to which the data contradicts the hypothesis. A small p-value implies that, *if* the hypothesis is true, the sample is very unusual. The smaller the p-value, the more strongly the evidence contradicts the hypothesis.

It is common to think that a hypothesis can be rejected only if the p-value is less than 5% (or some number). You should not think this way; the p-value summarizes the extent to which the data contradicts the hypothesis, and your particular application logic affects how you interpret it.

Worked example 10.9 *Are US and Japanese cars different*

At `http://www.itl.nist.gov/div898/handbook/eda/section3/eda3531.htm` you can find a dataset, published by NIST, giving measurements of miles per gallon for Japanese and US cars. Assess the evidence these two populations have the same mean MPG.

Solution: There are 249 measurements for Japanese cars, and 79 for US cars. The mean for Japanese cars is 20.1446, and for US cars is 30.4810. The standard error for Japanese cars is 0.4065, and for US cars is 0.6872. The value of the test statistic is

$$\frac{(\mathsf{mean}\,(\{x\}) - \mathsf{mean}\,(\{y\}))}{s_{ed}} = 10.33/0.7984 = 12.94$$

and the p-value is the probability of encountering a standard normal random variable of this value or greater. This is so close to zero I had trouble getting sensible numbers out of MATLAB; the evidence very strongly rejects this hypothesis. A version of this example, using the more complex two-sample t-test, is worked in the NIST/SEMATECH e-Handbook of Statistical Methods, at `http://www.itl.nist.gov/div898/handbook/`, as of 2013.

Worked example 10.10 *Do women and men have the same body temperature?*

Using the dataset of human temperatures at , assess the evidence against the hypothesis that gender 1 has the same mean temperature as gender 2

Solution: We compute the mean temperatures and standard errors shown in the table.

Gender:	1	2
Mean:	98.10	98.39
Std Error:	0.0867	0.0922

The null hypothesis is that these two are the same, so the test statistic is

$$\frac{\text{difference}}{\text{std error}} = \frac{98.39 - 98.10}{\sqrt{0.0867^2 + 0.0922^2}} = 2.285$$

and we must ask what is the probability of getting a number with absolute value this big, or bigger, from a normal distribution (two-sided test). This is 0.0223. The evidence in this data set does not support the null hypothesis.

10.3.3 Variants on the Basic Test

The test of procedure 3 is usually referred to as a **two-sided** test. This is because it computes

$$P(\{X > |s|\}) \cup P(\{X < -|s|\}).$$

This is the fraction of samples that would produce a value that is larger than $|s|$ or smaller than $-|s|$. In some cases, we can expect that only the larger (or smaller) value is of interest. This yields a **one-sided** test; the relevant expression is either

$$P(\{X > s\})$$

or

$$P(\{X < s\}).$$

Generally, it is more conservative to use a two-sided test, and one should do so unless there is a good reason not to. Very often, authors use one-sided tests because they result in smaller p-values, and small p-values are often a requirement for publication.

Z-Tests and T-Tests

Our procedure works because we were able to identify a random variable where: (a) the random variable represents the effect of drawing a sample of a population at random; (b) we knew the distribution of the random variable conditioned on the hypothesis; (c) we can evaluate the random variable for our particular sample. Now use the notation of procedure 3. In that procedure, we computed

$$P(\{X > |s|\}) \cup P(\{X < -|s|\}).$$

When we have a large sample, the sample mean is a normal random variable whose mean is the population mean and whose standard deviation is the standard error. This means that we know its distribution. In this case, the test is known as a **z-test**.

An important difficulty with this procedure is we assumed we know the standard error. As figure 10.2 suggested, estimating standard error with $\mathsf{std}\,(x)/\sqrt{k}$ works fine for large k, but is less successful for small k. Usually, practical advice suggests that one should use a Z-test only if the sample has at least 30 elements.

When the sample is small, the sample standard deviation is a poor estimate of the population standard deviation. The value of the sample mean that we compute minimizes the sample standard deviation, which means that the estimate tends to be a little too small. In turn, the standard error is a little too small, and there is slightly more probability that the sample mean is far from the population mean than the normal model allows for. This can be corrected for. Instead of using the standard deviation of the sample to form the standard error, we use

$$\sqrt{\frac{\sum_i (x_i - \mathsf{mean}\,(\{x_i\}))^2}{k-1}}.$$

When we test, instead of using the normal distribution to compute probabilities, we use **Student's t-distribution**. This is a family of probability distributions. There are two parameters; the observed value s, and the number of degrees of freedom.

The number of degrees of freedom is $k-1$ for our purposes. When the number of degrees of freedom is small, the t-distribution has rather heavier tails than the normal distribution, so the test takes into account that the standard error may be larger than we think (because the population standard deviation is larger than we expected). When the number of degrees of freedom is large, the t-distribution is very similar to the normal distribution. One can get p-values from tables, or by the Matlab function `ttest`.

10.4 χ^2 TESTS: IS DATA CONSISTENT WITH A MODEL?

Sometimes we have a model, and we would like to know whether the data is consistent with that model. For example, imagine we have a six-sided die. We throw it many times, and record which number comes up each time. We would like to know if the die is fair (i.e. is the data consistent with the model that the die is fair). It is highly unlikely that each face comes up the same number of times, even if the die is fair. Instead, there will be some variation in the frequencies observed; with what probability is that variation, or bigger, the result of chance effects?

As another example, we decide that the number of calls by a telemarketer in each hour is distributed with a Poisson distribution. We don't know the intensity. We could collect call data, and use maximum likelihood to determine the intensity. Once we have the best estimate of the intensity, we still want to know whether the model is consistent with the data.

In each case, the model predicts the frequencies of events. For the six-sided die case, the model tells us how often we expect to see each side. For the call case, the model predicts how often we would see no calls, one call, two calls, three calls, etc. in each hour. To tell whether the model fits the data, we need to compare the frequencies we observed with theoretical frequencies.

The appropriate test is a χ^2 (say "khi-squared") test. Assume we have a set of disjoint events $\mathcal{E}_1, \ldots, \mathcal{E}_k$ which cover the space of outcomes (i.e. any outcome lies in one of these events). Assume we perform k experiments, and record the number of times each event occurs. We have a null hypothesis regarding the probability of events. We can take the probability of each event and multiply by k to get a frequency under the null hypothesis. Now write $f_o(\mathcal{E}_i)$ for the observed frequency of event i; $f_t(\mathcal{E}_i)$ for the theoretical frequency of the event under the null hypothesis. We form the statistic

$$\sum_i \frac{(f_o(\mathcal{E}_i) - f_t(\mathcal{E}_i)^2}{f_t(\mathcal{E}_i)}$$

which compares the observed and actual frequency of events. It turns out that this statistic has a distribution very close to a known form, called the χ^2 distribution, as long as each count is 5 or more. The distribution has two parameters; the statistic, and the number of degrees of freedom. The number of degrees of freedom to use for a straightforward test is $k-1$; if one has to estimate a total of p parameters for the null hypothesis, this number is $k-p-1$.

After this, things follow the usual recipe. We compute the statistic; we then look at tables, or use the matlab function `chi2cdf`, to find the probability that the statistic takes this value or greater under the null hypothesis. If this is small, then we reject the null hypothesis.

Worked example 10.11 χ^2 *test for dice*

I throw a die 100 times. I record the outcomes, in the table below. Is this a fair die?

face	count
1	46
2	13
3	12
4	11
5	9
6	9

Solution: The expected frequency is 100/6 for each face. The χ^2 statistic has the value 62.7, and there are 5 degrees of freedom. We get the significance as `1-chi2cdf(62.7, 5)`, which is (basically) 3e-12. You would have to run this experiment 3e11 times to see a table as skewed as this once, by chance. The die is not fair. Notice the most important bit of this example, which is how to get the number out of matlab.

Worked example 10.12 *Is swearing Poisson?*

A famously sweary politician gives a talk. You listen to the talk, and for each of 30 intervals 1 minute long, you record the number of swearwords. You record this as a histogram (i.e. you count the number of intervals with zero swear words, with one, etc.).

no. of swear words	no. of intervals
0	13
1	9
2	8
3	5
4	5

The null hypothesis is that the politician's swearing is Poisson distributed, with intensity (λ) one. Can you reject this null hypothesis?

Solution: If the null hypothesis is true, then the probability of getting n swear words in a fixed length interval would be $\frac{\lambda^n e^{-\lambda}}{n!}$. There are 10 intervals, so the theoretical frequencies are 10 times the following probabilities

number of swear words	probability
0	0.368
1	0.368
2	0.184
3	0.061
4	0.015

so the χ^2 statistic takes the value 243.1 and there are 4 degrees of freedom. The significance `1-chi2cdf(243.1, 4)` is indistinguishable from zero by matlab, so you can firmly reject the null hypothesis. Of course, the intensity may be wrong; but we don't know how to deal with that, and will have to discuss that question in the next chapter.

Worked example 10.13 *Is gender 2 temperature normal?*

Recall we used body temperature data for two genders in earlier examples. The gender 2 data looked as though it might not be normal. Use a χ^2 test to tell whether it is normal with mean 98.4^o and standard deviation 0.743 or not.

Solution: The null hypothesis is that the data is normal with mean 98.4^o and standard deviation 0.743. We break up the range into five buckets (less than $97.65 = 98.4 - 0.74$; between 97.65 and $98 = 98.4 - 0.743/2$; between 98 and $98.76 = 98.4 + 0.743/2$; and greater than 99.14). With a little work with error functions, we get that the theoretical frequency in each bucket under the null hypothesis is $(10.3126; 9.7423; 24.8901; 9.7423; 10.3126)$. The actual frequencies are $(7; 13; 26; 12; 7)$. The χ^2 statistic is about 4e3, and the significance level is essentially zero. This data isn't normal with the parameters given (though it might be normal with a different mean and a different standard deviation). Looking at the frequencies suggests the problem; there are far too few temperatures far away from the mean for this to be normal, and far too many in the center bucket.

10.5 WHAT YOU SHOULD REMEMBER - NEED SOMETHING

PROBLEMS

Confidence Intervals

10.1. The Weight of Mice You wish to estimate the average weight of a mouse. You obtain 10 mice, sampled uniformly at random and with replacement from the mouse population. Their weights are $21, 23, 27, 19, 17, 18, 20, 15, 17, 22$ grams respectively.
(a) Give a 68% confidence interval for the weight of a mouse, from this data.
(b) Give a 99% confidence interval for the weight of a mouse, from this data.

10.2. The Weight of Rats You wish to estimate the average weight of a pet rat. You obtain 40 rats (easily and cheaply done; keep them, because they're excellent pets), sampled uniformly at random and with replacement from the pet rat population. The mean weight is 340 grams, with a standard deviation of 75 grams.
(a) Give a 68% confidence interval for the weight of a pet rat, from this data.
(b) Give a 99% confidence interval for the weight of a pet rat, from this data.

Hypothesis Testing

10.3. Yet more Mouse-weighing I claim the average weight of a mouse is 25 grams. You decide to evaluate the evidence in support of this claim. You obtain 10 mice, sampled uniformly at random and with replacement from the mouse population. Their weights are $21, 23, 27, 19, 17, 18, 20, 15, 17, 22$ grams respectively. Does the evidence support my claim? to what extent? Why?

10.4. How big are Parktown Prawns? The Parktown prawn is an impressively repellent large insect, common in Johannesburg (look them up on the Web).

I claim that their average length is 10cm. You collect 100 Parktown prawns (this will take about 10 mins, in the right places in Johannesburg; more difficult from here). The mean length of these prawns is 7cm. The standard deviation is 1cm. Assess the evidence against my claim.

10.5. Two Populations of Rats Zucker rats are specially bred to have curious weight properties, related to their genetics (look them up on the Web). You measure 30 lean Zucker rats, obtaining an average weight of 500 grams with a standard deviation of 50 grams. You measure 20 fatty Zucker rats, obtaining an average weight of 1000 grams with a standard deviation of 100 grams. Assess the evidence against the claim that these populuations have the same weight.

10.6. Male and Female pet Rats Zucker rats are specially bred to have curious weight properties, related to their genetics (look them up on the Web). You measure 25 female pet rats, obtaining an average weight of 300 grams with a standard deviation of 30 grams. You measure 20 male pet rats, obtaining an average weight of 400 grams with a standard deviation of 100 grams. Assess the evidence against the claim that these populations have the same weight.

CHAPTER 11

Extracting Important Relationships in High Dimensions

Chapter 4 described methods to explore the relationship between two elements in a dataset. We could extract a pair of elements and construct various plots. For vector data, we could also compute the correlation between different pairs of elements. But if each data item is d-dimensional, there could be a lot of pairs to deal with.

We will think of our dataset as a collection of d dimensional vectors. It turns out that there are easy generalizations of our summaries. However, is hard to plot d-dimensional vectors. We need to find some way to make them fit on a 2-dimensional plot. Some simple methods can offer insights, but to really get what is going on we need methods that can at all pairs of relationships in a dataset in one go.

These methods visualize the dataset as a "blob" in a d-dimensional space. Many such blobs are flattened in some directions, because components of the data are strongly correlated. Finding the directions in which the blobs are flat yields methods to compute lower dimensional representations of the dataset.

11.1 SUMMARIES AND SIMPLE PLOTS

In this chapter, we assume that our data items are vectors. This means that we can add and subtract values and multiply values by a scalar without any distress. This is an important assumption, but it doesn't necessarily mean that data is continuous (for example, you can meaningfully add the number of children in one family to the number of children in another family). It does rule out a lot of discrete data. For example, you can't add "sports" to "grades" and expect a sensible answer.

Notation: Our data items are vectors, and we write a vector as $\mathbf{x}$. The data items are d-dimensional, and there are N of them. The entire data set is $\{\mathbf{x}\}$. When we need to refer to the i'th data item, we write $\mathbf{x}_i$. We write $\{\mathbf{x}_i\}$ for a new dataset made up of N items, where the i'th item is $\mathbf{x}_i$. If we need to refer to the j'th component of a vector $\mathbf{x}_i$, we will write $x_i^{(j)}$ (notice this isn't in bold, because it is a component not a vector, and the j is in parentheses because it isn't a power). Vectors are always column vectors.

11.1.1 The Mean

For one-dimensional data, we wrote

$$\text{mean}\left(\{x\}\right) = \frac{\sum_i x_i}{N}.$$

This expression is meaningful for vectors, too, because we can add vectors and divide by scalars. We write

$$\text{mean}\left(\{\mathbf{x}\}\right) = \frac{\sum_i \mathbf{x}_i}{N}$$

and call this the mean of the data. Notice that each component of $\text{mean}\left(\{\mathbf{x}\}\right)$ is the mean of that component of the data. There is not an easy analogue of the median, however (how do you order high dimensional data?) and this is a nuisance. Notice that, just as for the one-dimensional mean, we have

$$\text{mean}\left(\{\mathbf{x} - \text{mean}\left(\{\mathbf{x}\}\right)\}\right) = 0$$

(i.e. if you subtract the mean from a data set, the resulting data set has zero mean).

11.1.2 Parallel Plots

Parallel plots can sometimes reveal information, particularly when the dimension of the dataset is low. To construct a parallel plot, you compute a normalized representation of each component of each data item. The component is normalized by translating and scaling so that the minimum value over the dataset is zero, and the maximum value over the dataset is one. Now write the i'th normalised data item as $(n_1, n_2, \ldots, n_d)$. For this data item, you plot a broken line joining $(1, n_1)$ to $(2, n_2)$ to $(3, n_3$, etc. These plots are superimposed on one another. In the case of the bodyfat dataset, this yields the plot of figure 11.1.

Some structures in the parallel plot are revealing. Outliers often stick out (in figure 11.1, it's pretty clear that there's a data point with a very low height value, and also one with a very large weight value). Outliers affect the scaling, and so make other structures difficult to spot. I have removed them for figure 11.2. In this figure, you can see that two negatively correlated components next to one another produce a butterfly like shape (bodyfat and density). In this plot, you can also see that there are still several data points that are very different from others (two data items have ankle values that are very different from the others, for example).

11.1.3 Using Covariance to encode Variance and Correlation

Variance, standard deviation and correlation can each be seen as an instance of a more general operation on data. Assume that we have two one dimensional data sets $\{x\}$ and $\{y\}$. Then we can define the **covariance** of $\{x\}$ and $\{y\}$.

Definition: 11.1 *Covariance*

Assume we have two sets of N data items, $\{x\}$ and $\{y\}$. We compute the covariance by

$$\text{cov}\left(\{x\}, \{y\}\right) = \frac{\sum_i (x_i - \text{mean}\left(\{x\}\right))(y_i - \text{mean}\left(\{y\}\right))}{N}$$

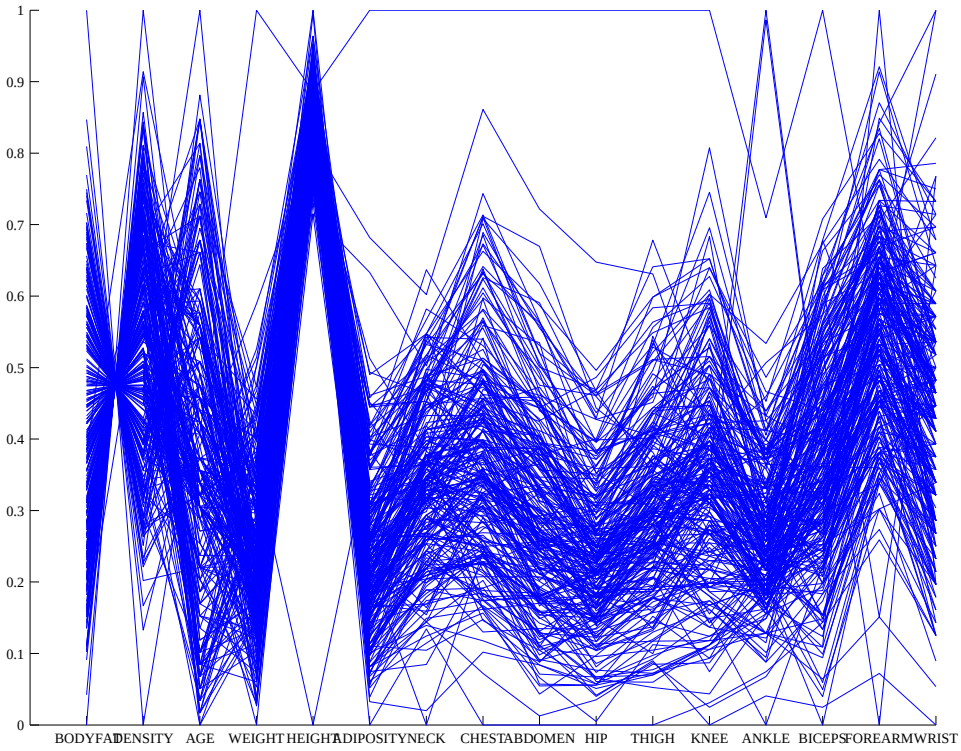

FIGURE 11.1: *A parallel plot of the bodyfat dataset, including all data points. I have named the components on the horizontal axis. It is easy to see that large values of bodyfat correspond to small values of density, and vice versa. Notice that one datapoint has height very different from all others; similarly, one datapoint has weight very different from all others.*

Covariance measures the tendency of corresponding elements of $\{x\}$ and of $\{y\}$ to be larger than (resp. smaller than) the mean. Just like mean, standard deviation and variance, covariance can refer either to a property of a dataset (as in the definition here) or a particular expectation (as in chapter 6). The correspondence is defined by the order of elements in the data set, so that x_1 corresponds to y_1, x_2 corresponds to y_2, and so on. If $\{x\}$ tends to be larger (resp. smaller) than its mean for data points where $\{y\}$ is also larger (resp. smaller) than its mean, then the covariance should be positive. If $\{x\}$ tends to be larger (resp. smaller) than its mean for data points where $\{y\}$ is smaller (resp. larger) than its mean, then the covariance should be negative.

From this description, it should be clear we have seen examples of covariance already. Notice that

$$\mathsf{std}\left(x\right)^2 = \mathsf{var}\left(\{x\}\right) = \mathsf{cov}\left(\{x\},\{x\}\right)$$

which you can prove by substituting the expressions. Recall that variance measures

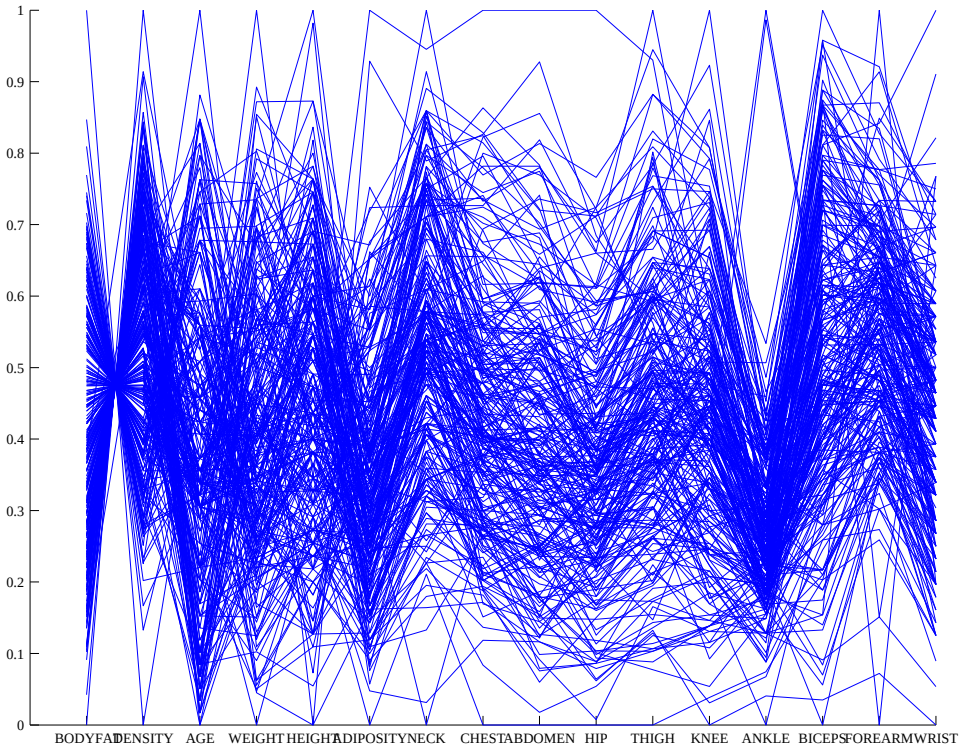

FIGURE 11.2: *A plot with those data items removed, so that those components are renormalized. Two datapoints have rather distinct ankle measurements. Generally, you can see that large knees go with large ankles and large biceps (the v structure).*

the tendency of a dataset to be different from the mean, so the covariance of a dataset with itself is a measure of its tendency not to be constant.

More important, notice that

$$\mathsf{corr}\left(\{(x,y)\}\right) = \frac{\mathsf{cov}\left(\{x\},\{y\}\right)}{\sqrt{\mathsf{cov}\left(\{x\},\{x\}\right)}\sqrt{\mathsf{cov}\left(\{y\},\{y\}\right)}}.$$

This is occasionally a useful way to think about correlation. It says that the correlation measures the tendency of $\{x\}$ and $\{y\}$ to be larger (resp. smaller) than their means for the same data points, *compared to* how much they change on their own.

Working with covariance (rather than correlation) allows us to unify some ideas. In particular, for data items which are d dimensional vectors, it is straightforward to compute a single matrix that captures all covariances between all pairs of components — this is the **covariance matrix**.

Definition: 11.2 *Covariance Matrix*

The covariance matrix is:

$$\mathsf{Covmat}\left(\{\mathbf{x}\}\right) = \frac{\sum_i (\mathbf{x}_i - \mathsf{mean}\left(\{\mathbf{x}\}\right))(\mathbf{x}_i - \mathsf{mean}\left(\{\mathbf{x}\}\right))^T}{N}$$

Notice that it is quite usual to write a covariance matrix as Σ, and we will follow this convention.

Properties of the Covariance Matrix Covariance matrices are often written as Σ, whatever the dataset (you get to figure out precisely which dataset is intended, from context). Generally, when we want to refer to the j, k'th entry of a matrix $\mathcal{A}$, we will write $\mathcal{A}_{jk}$, so Σ_{jk} is the covariance between the j'th and k'th components of the data.

- The j, k'th entry of the covariance matrix is the covariance of the j'th and the k'th components of $\mathbf{x}$, which we write $\mathsf{cov}\left(\left\{x^{(j)}\right\}, \left\{x^{(k)}\right\}\right)$.
- The j, j'th entry of the covariance matrix is the variance of the j'th component of $\mathbf{x}$.
- The covariance matrix is symmetric.
- The covariance matrix is always positive semi-definite; it is positive definite, *unless* there is some vector $\mathbf{a}$ such that $\mathbf{a}^T(\mathbf{x}_i - \mathsf{mean}\left(\{\mathbf{x}_i\}\right) = 0$ for all i.

Proposition:

$$\mathsf{Covmat}\left(\{\mathbf{x}\}\right)_{jk} = \mathsf{cov}\left(\left\{x^{(j)}\right\}, \left\{x^{(k)}\right\}\right)$$

Proof: Recall

$$\mathsf{Covmat}\left(\{\mathbf{x}\}\right) = \frac{\sum_i (\mathbf{x}_i - \mathsf{mean}\left(\{\mathbf{x}\}\right))(\mathbf{x}_i - \mathsf{mean}\left(\{\mathbf{x}\}\right))^T}{N}$$

and the j, k'th entry in this matrix will be

$$\frac{\sum_i (x_i^{(j)} - \mathsf{mean}\left(\left\{x^{(j)}\right\}\right))(x_i^{(k)} - \mathsf{mean}\left(\left\{x^{(k)}\right\}\right))^T}{N}$$

which is $\mathsf{cov}\left(\left\{x^{(j)}\right\}, \left\{x^{(k)}\right\}\right)$.

Proposition:

$$\text{Covmat}\left(\{\mathbf{x}_i\}\right)_{jj} = \Sigma_{jj} = \text{var}\left(\left\{x^{(j)}\right\}\right)$$

Proof:

$$\begin{aligned}\text{Covmat}\left(\{\mathbf{x}\}\right)_{jj} &= \text{cov}\left(\left\{x^{(j)}\right\}, \left\{x^{(j)}\right\}\right) \\ &= \text{var}\left(\left\{x^{(j)}\right\}\right)\end{aligned}$$

Proposition:

$$\text{Covmat}\left(\{\mathbf{x}\}\right) = \text{Covmat}\left(\{\mathbf{x}\}\right)^T$$

Proof: We have

$$\begin{aligned}\text{Covmat}\left(\{\mathbf{x}\}\right)_{jk} &= \text{cov}\left(\left\{x^{(j)}\right\}, \left\{x^{(k)}\right\}\right) \\ &= \text{cov}\left(\left\{x^{(k)}\right\}, \left\{x^{(j)}\right\}\right) \\ &= \text{Covmat}\left(\{\mathbf{x}\}\right)_{kj}\end{aligned}$$

Proposition: *Write* $\Sigma = \mathsf{Covmat}(\{\mathbf{x}\})$. *If there is no vector* $\mathbf{a}$ *such that* $\mathbf{a}^T(\mathbf{x}_i - \mathsf{mean}(\{\mathbf{x}\})) = 0$ *for all* i, *then for any vector* $\mathbf{u}$, *such that* $\|\mathbf{u}\| > 0$,

$$\mathbf{u}^T\Sigma\mathbf{u} > 0.$$

If there is such a vector $\mathbf{a}$, *then*

$$\mathbf{u}^T\Sigma\mathbf{u} \geq 0.$$

Proof: We have

$$\begin{aligned}\mathbf{u}^T\Sigma\mathbf{u} &= \frac{1}{N}\sum_i \left[\mathbf{u}^T(\mathbf{x}_i - \mathsf{mean}(\{\mathbf{x}\}))\right]\left[(\mathbf{x}_i - \mathsf{mean}(\{\mathbf{x}\}))^T\mathbf{u}\right] \\ &= \frac{1}{N}\sum_i \left[\mathbf{u}^T(\mathbf{x}_i - \mathsf{mean}(\{\mathbf{x}\}))\right]^2.\end{aligned}$$

Now this is a sum of squares. If there is some $\mathbf{a}$ such that $\mathbf{a}^T(\mathbf{x}_i - \mathsf{mean}(\{\mathbf{x}\})) = 0$ for every i, then the covariance matrix must be positive semidefinite (because the sum of squares could be zero in this case). Otherwise, it is positive definite, because the sum of squares will always be positive.

11.2 BLOB ANALYSIS OF HIGH-DIMENSIONAL DATA

When we plotted histograms, we saw that mean and variance were a very helpful description of data that had a unimodal histogram. If the histogram had more than one mode, one needed to be somewhat careful to interpret the mean and variance; in the pizza example, we plotted diameters for different manufacturers to try and see the data as a collection of unimodal histograms.

Generally, mean and covariance are a good description of data that lies in a "blob" (Figure 11.3). You might not believe that this is a technical term, but it's quite widely used. This is because mean and covariance supply a natural coordinate system in which to interpret the blob. Mean and covariance are less useful as descriptions of data that forms multiple blobs (Figure 11.3). In chapter 1, we discuss methods to model data that forms multiple blobs, or other shapes that we will interpret as a set of blobs. But many datasets really are single blobs, and we concentrate on such data here. The way to understand a blob is to think about the coordinate transformations that place a blob into a particularly convenient form.

11.2.1 Understanding Blobs with Scatterplot Matrices - CLEANUP

Plotting high dimensional data is tricky. One strategy that is very useful when there aren't too many dimensions is to use a scatterplot matrix. To build one,

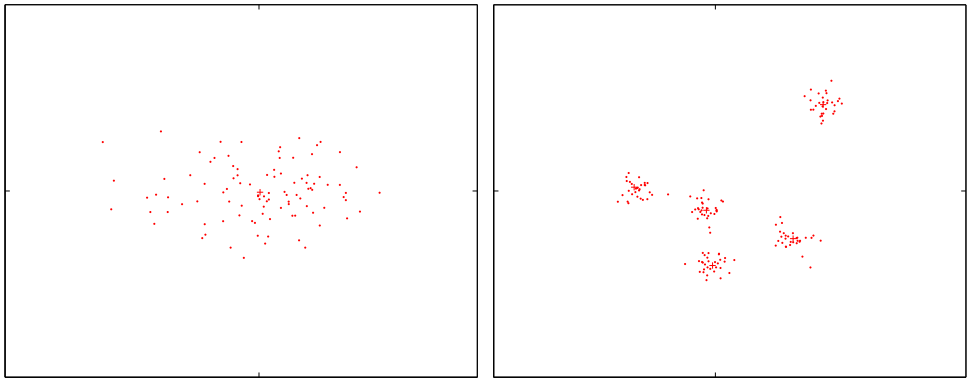

FIGURE 11.3: *On the* **left**, *a "blob" in two dimensions. This is a set of data points that lie somewhat clustered around a single center, given by the mean. I have plotted the mean of these data points with a '+'. On the* **right**, *a data set that is best thought of as a collection of five blobs. I have plotted the mean of each with a '+'. We could compute the mean and covariance of this data, but it would be less revealing than the mean and covariance of a single blob. In chapter 1, I will describe automatic methods to describe this dataset as a series of blobs.*

you lay out scatterplots for each pair of variables in a matrix. On the diagonal, you name the variable that is the vertical axis for each plot in the row, and the horizontal axis in the column. This sounds more complicated than it is; look at the example of figure 11.4, which shows a scatterplot matrix for four of the variables in the height weight dataset of `http://www2.stetson.edu/~jrasp/data.htm`; look for bodyfat.xls at that URL). This is originally a 16-dimensional dataset, but a 16 by 16 scatterplot matrix is squashed and hard to interpret.

What is nice about this kind of plot is that it's quite easy to spot correlations between pairs of variables, though you do need to take into account the coordinates have not been normalized. For figure 11.4, you can see that weight and adiposity appear to show quite strong correlations, but weight and age are pretty weakly correlated. Height and age seem to have a low correlation. It is also easy to visualize unusual data points. Usually one has an interactive process to do so — you can move a "brush" over the plot to change the color of data points under the brush. To show what might happen, figure 11.5 shows a scatter plot matrix with some points shown as circles. Notice how they lie inside the "blob" of data in some views, and outside in others. This is an effect of projection.

UC Irvine keeps a large repository of datasets that are important in machine learning. You can find the repository at `http://archive.ics.uci.edu/ml/index.html`. Figures 11.6 and 11.7 show visualizations of a famous dataset to do with the botanical classification of irises.

Figures **??**, **??** and 11.14 show visualizations of another dataset to do with forest fires in Portugal, also from the UC Irvine repository (look at `http://archive.ics.uci.edu/ml/datasets/Forest+Fires`). In this dataset, there are a variety of measurements of location, time, temperature, etc. together with the area burned

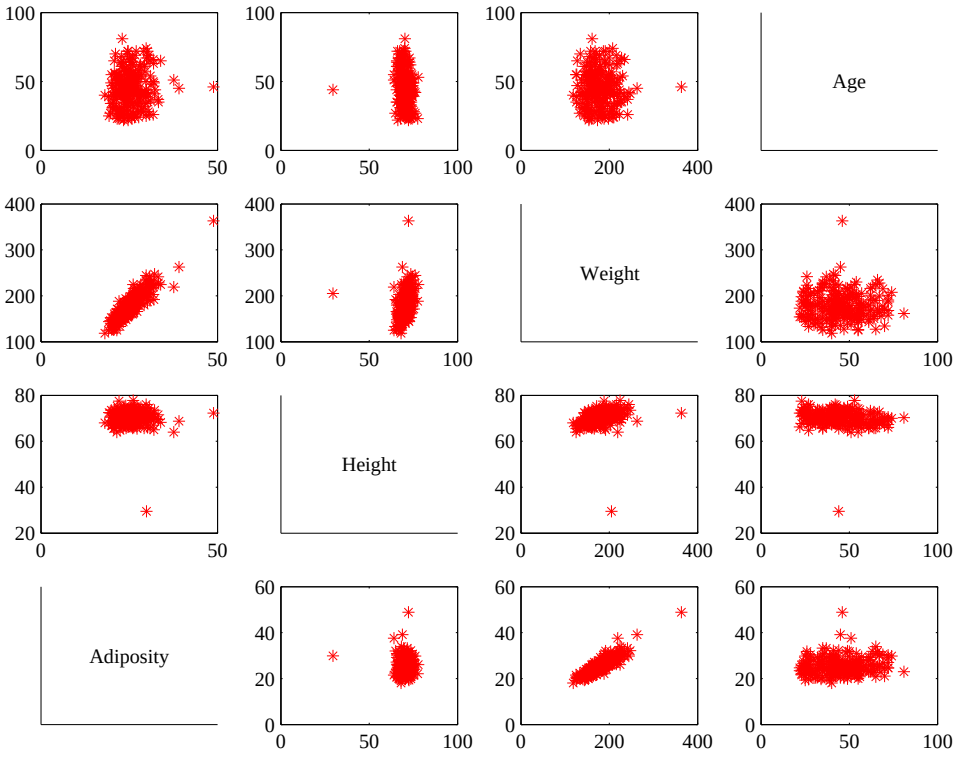

FIGURE 11.4: *This is a scatterplot matrix for four of the variables in the height weight dataset of* `http://www2.stetson.edu/~jrasp/data.htm`. *Each plot is a scatterplot of a pair of variables. The name of the variable for the horizontal axis is obtained by running your eye down the column; for the vertical axis, along the row. Although this plot is redundant (half of the plots are just flipped versions of the other half), that redundancy makes it easier to follow points by eye. You can look at a column, move down to a row, move across to a column, etc. Notice how you can spot correlations between variables and outliers (the arrows).*

by a wildfire. It would be nice to know what leads to large fires, and a visualization is the place to start. Many fires are tiny (or perhaps there was no area measurement?) and so many values of the area are zero. I found it helpful to take the log of area, and then to divide the values of the logarithm into seven categories. I ignored the first four variables, because I didn't think they'd be too important. **Exercise:** was I right? I made two scatterplot matrices, because an eight by eight matrix is too big to view. Generally, this visualization suggests that it would be hard to predict the size of a fire from these variables.

We can combine tools to analyze datasets. In the UC Irvine repository, you can find a dataset related to heart disease (look at `http://archive.ics.uci.edu/ml/datasets/Heart+Disease`). There are a variety of versions of the dataset; I used the version in the file "processed.cleveland.data". This contains a set of 14

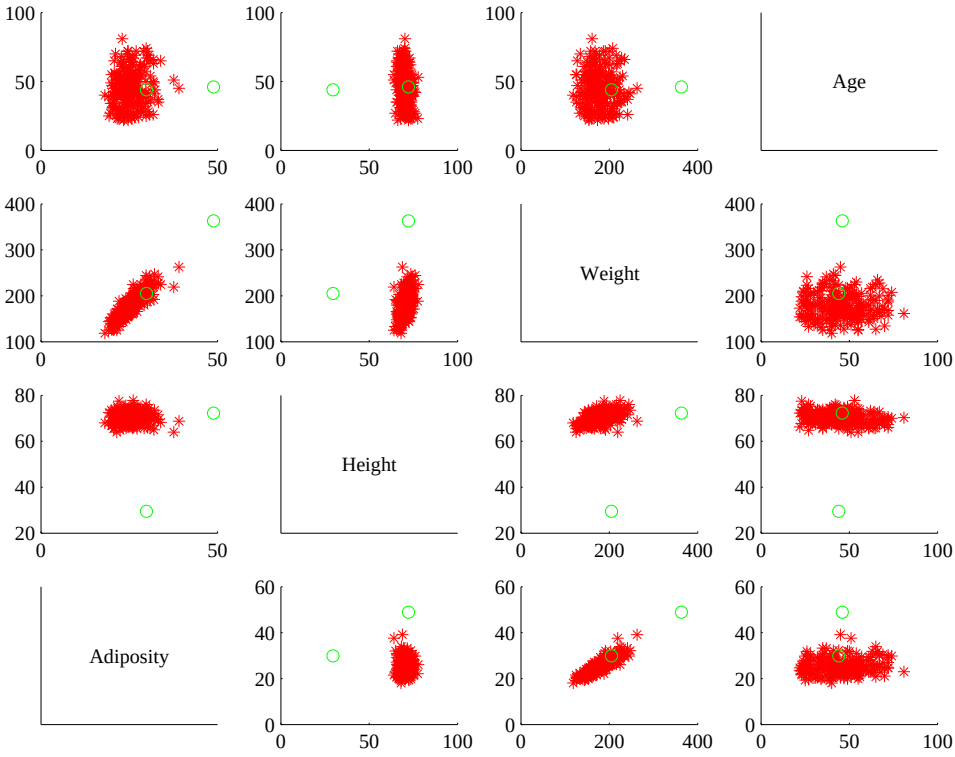

FIGURE 11.5: *You should compare this figure with figure 11.4. I have marked two data points with circles in this figure; notice that in some panels these are far from the rest of the data, in others close by. A "brush" in an interactive application can be used to mark data like this to allow a user to explore a dataset.*

features describing individuals under study. The 14'th is a measure of heart disease. What can we learn from this dataset?

The first thing to notice is that many of the variables are categorical. It is natural to make some mosaic plots to visualize what is happening. I quantized the age to five levels (0-20, 20-40, etc.), and quantized the measure of heart disease to two levels (no disease and disease) to simplify the plot. Figure **??** shows a mosaic plot of the result.

11.2.2 Transforming High Dimensional Data

Assume we apply an affine transformation to our data set $\{\mathbf{x}\}$, to obtain a new dataset $\{\mathbf{u}\}$, where $\mathbf{u}_i = \mathcal{A}\mathbf{x}_i + \mathbf{b}$. Here $\mathcal{A}$ is any matrix (it doesn't have to be square, or symmetric, or anything else; it just has to have second dimension d). It is easy to compute the mean and covariance of $\{\mathbf{u}\}$. We have

$$\begin{aligned} \text{mean}(\{\mathbf{u}\}) &= \text{mean}(\{\mathcal{A}\mathbf{x} + \mathbf{b}\}) \\ &= \mathcal{A}\text{mean}(\{\mathbf{x}\}) + \mathbf{b}, \end{aligned}$$

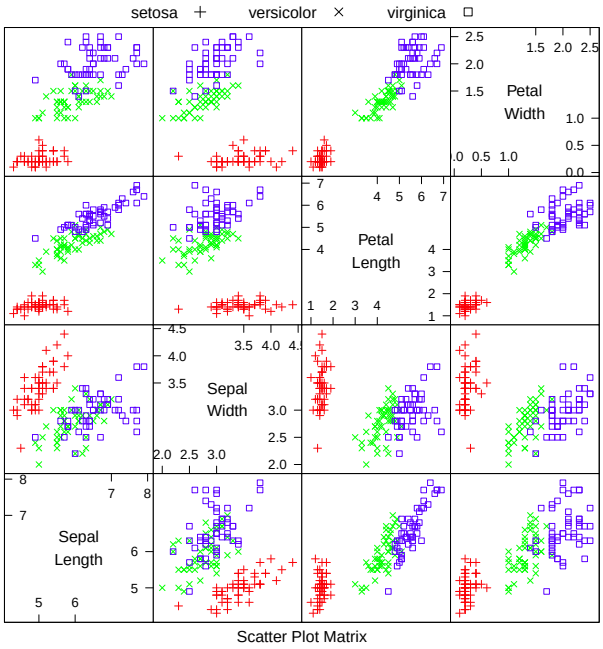

FIGURE 11.6: *This is a scatterplot matrix for the famous Iris data, originally due to ***. There are four variables, measured for each of three species of iris. I have plotted each species with a different marker. You can see from the plot that the species cluster quite tightly, and are different from one another. R code for this plot is on the website.*

so you get the new mean by multiplying the original mean by $\mathcal{A}$ and adding $\mathbf{b}$.

The new covariance matrix is easy to compute as well. We have:

$$\begin{aligned}
\text{Covmat}(\{\mathbf{u}\}) &= \text{Covmat}(\{\mathcal{A}\mathbf{x}+\mathbf{b}\}) \\
&= \frac{\sum_i (\mathbf{u}_i - \text{mean}(\{\mathbf{u}\}))(\mathbf{u}_i - \text{mean}(\{\mathbf{u}\}))^T}{N} \\
&= \frac{\sum_i (\mathcal{A}\mathbf{x}_i + \mathbf{b} - \mathcal{A}\text{mean}(\{\mathbf{x}\}) - \mathbf{b})(\mathcal{A}\mathbf{x}_i + \mathbf{b} - \mathcal{A}\text{mean}(\{\mathbf{x}\}) - \mathbf{b})^T}{N} \\
&= \frac{\mathcal{A}\sum_i (\mathbf{x}_i - \text{mean}(\{\mathbf{x}\}))(\mathbf{x}_i - \text{mean}(\{\mathbf{x}\}))^T\mathcal{A}^T}{N} \\
&= \mathcal{A}\text{Covmat}(\{\mathbf{x}\})\mathcal{A}^T.
\end{aligned}$$

11.2.3 Transforming Blobs

The trick to interpreting high dimensional data is to use the mean and covariance to understand the blob. Figure 11.15 shows a two-dimensional data set. Notice that there is obviously some correlation between the x and y coordinates (it's a diagonal blob), and that neither x nor y has zero mean. We can easily compute the mean and subtract it from the data points, and this translates the blob so that the

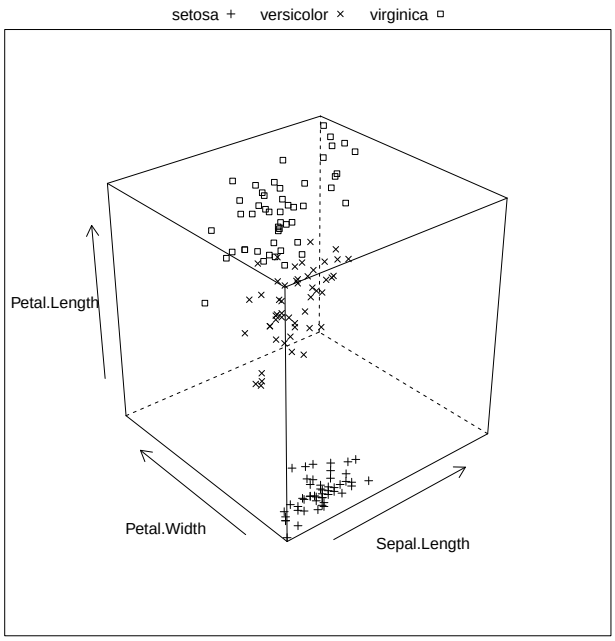

FIGURE 11.7: *This is a 3D scatterplot for the famous Iris data, originally due to ***. I have chosen three variables from the four, and have plotted each species with a different marker. You can see from the plot that the species cluster quite tightly, and are different from one another. R code for this plot is on the website.*

origin is at the center (Figure 11.15). In coordinates, this means we compute the new dataset $\{\mathbf{u}\}$ from the old dataset $\{\mathbf{x}\}$ by the rule $\mathbf{u}_i = \mathbf{x}_i - \mathsf{mean}\,(\{\mathbf{x}\})$. This new dataset has been translated so that the mean is zero.

Once this blob is translated (Figure 11.16, left), we can rotate it as well. It is natural to try to rotate the blob so that there is no correlation between distinct pairs of dimensions. We can do so by diagonalizing the covariance matrix. In particular, let $\mathcal{U}$ be the matrix formed by stacking the eigenvectors of $\mathsf{Covmat}\,(\{\mathbf{x}\})$ into a matrix (i.e. $\mathcal{U} = [\mathbf{v}_1, \ldots, \mathbf{v}_d]$, where $\mathbf{v}_j$ are eigenvectors of the covariance matrix). We now form the dataset $\{\mathbf{n}\}$, using the rule

$$\mathbf{n}_i = \mathcal{U}^T \mathbf{u}_i = \mathcal{U}^T (\mathbf{x}_i - \mathsf{mean}\,(\{\mathbf{x}\})).$$

The mean of this new dataset is clearly $\mathbf{0}$. The covariance of this dataset is

$$\begin{aligned} \mathsf{Covmat}\,(\{\mathbf{n}\}) &= \mathsf{Covmat}\left(\{\mathcal{U}^T\mathbf{x}\}\right) \\ &= \mathcal{U}^T \mathsf{Covmat}\,(\{\mathbf{x}\})\mathcal{U} \\ &= \Lambda, \end{aligned}$$

where Λ is a diagonal matrix of eigenvalues of $\mathsf{Covmat}\,(\{\mathbf{x}\})$. Remember that, in describing diagonalization, we adopted the convention that the eigenvectors of the

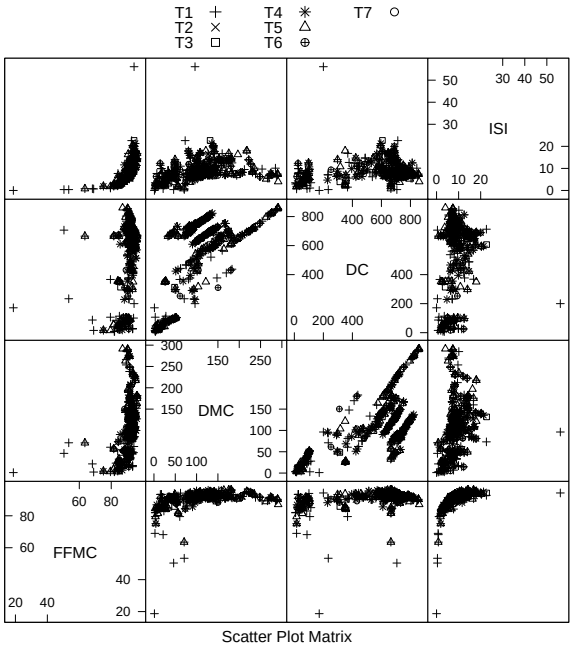

FIGURE 11.8: *This is a scatterplot matrix for the fire dataset from the UC Irvine repository. The smallest area fire is 'T1', and the largest is 'T7'; each is plotted with a different marker. These plots show severity of the fire, plotted against variables 5-8 of the dataset. You should notice that there isn't much separation between the markers. It might be very hard to predict the severity of a fire from these variables. R code for this plot is on the website.*

matrix being diagonalized were ordered so that the eigenvalues are sorted in descending order along the diagonal of Λ. We now have two very useful facts about $\{\mathbf{n}\}$: (a) every pair of distinct components has covariance zero, and so has correlation zero; (b) the first component has the highest variance, the second component has the second highest variance, and so on. We can rotate and translate any blob into a coordinate system that has these properties. *In this coordinate system*, we can describe the blob simply by giving the variances of each component — the covariances are zero.

Translating a blob of data doesn't change the scatterplot matrix in any interesting way (the axes change, but the picture doesn't). Rotating a blob produces really interesting results, however. Figure 11.18 shows the dataset of figure 11.4, translated to the origin and rotated to diagonalize it. Now we do not have names for each component of the data (they're linear combinations of the original components), but each pair is now not correlated. This blob has some interesting shape features. Figure 11.18 shows the gross shape of the blob best. Each panel of this figure has the same scale in each direction. You can see the blob extends about 80 units in direction 1, but only about 15 units in direction 2, and much less in the

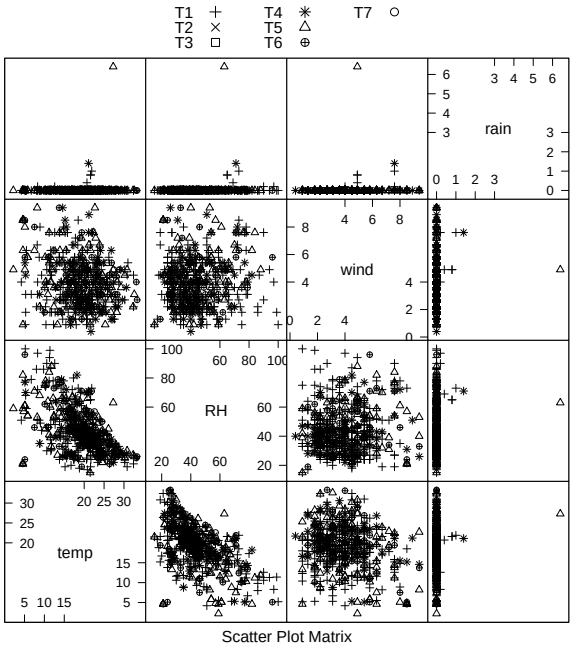

FIGURE 11.9: *This is a scatterplot matrix for the fire dataset from the UC Irvine repository. The smallest area fire is 'T1', and the largest is 'T7'; each is plotted with a different marker. These plots show severity of the fire, plotted against variables 9-12 of the dataset. You should notice that there isn't much separation between the markers. It might be very hard to predict the severity of a fire from these variables. R code for this plot is on the website.*

other two directions. You should think of this blob as being rather cigar-shaped; it's long in one direction, but there isn't much in the others. The cigar metaphor isn't perfect because there aren't any 4 dimensional cigars, but it's helpful. You can think of each panel of this figure as showing views down each of the four axes of the cigar.

Now look at figure **??**. This shows the same rotation of the same blob of data, but now the scales on the axis have changed to get the best look at the detailed shape of the blob. First, you can see that blob is a little curved (look at the projection onto direction 2 and direction 4). There might be some effect here worth studying. Second, you can see that some points seem to lie away from the main blob. I have plotted each data point with a dot, and the interesting points with a number. These points are clearly special in some way.

We could now scale the data in this new coordinate system so that all the variances are either one (if there is any variation in that direction) or zero (directions where the data doesn't vary — these occur only if some directions are functions of others). Figure 11.17 shows the final scaling. The result is a standard blob. Our approach applies to any dimension — I gave 2D figures because they're much easier

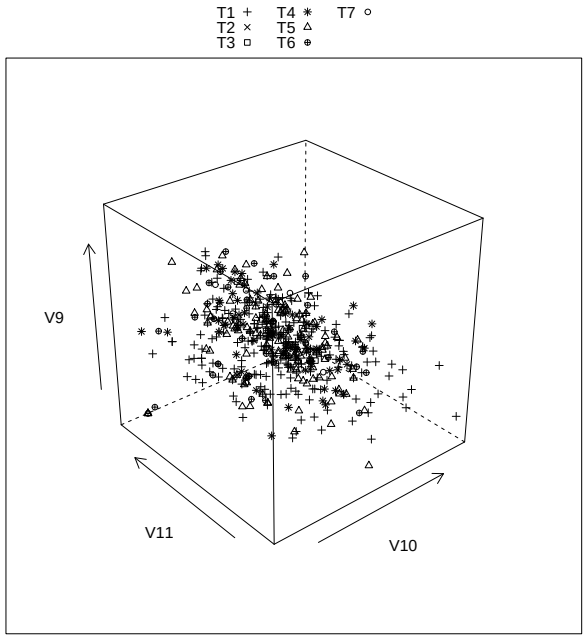

FIGURE 11.10: *This is a 3D scatterplot for the fire dataset from the UC Irvine repository. The smallest area fire is 'T1', and the largest is 'T7'; each is plotted with a different marker. These plots show severity of the fire, plotted against variables 9-11 of the dataset. You should notice that there isn't much separation between the markers. It might be very hard to predict the severity of a fire from these variables. R code for this plot is on the website.*

to understand. There is a crucial point here: we can reduce any blob of data, in any dimension, to a standard blob of that dimension. All blobs are the same, except for some stretching, some rotation, and some translation. This is why blobs are so well-liked.

11.2.4 Whitening Data

It is sometimes useful to actually reduce a dataset to a standard blob. Doing so is known as **whitening the data** (for reasons I find obscure). This can be a sensible thing to do when we don't have a clear sense of the relative scales of the components of each data vector. For example, if we have a dataset where one component ranges from 1e5 to 2e5, and the other component ranges from -1e-5 to 1e-5, we are likely to face numerical problems in many computations (adding small numbers to big numbers is often unwise). Often, this kind of thing follows from a poor choice of units, rather than any kind of useful property of the data. In such a case, it could be quite helpful to whiten the data. Another reason to whiten the data might be that we know relatively little about the meaning of each component. In this case, the original choice of coordinate system was somewhat arbitrary anyhow, and

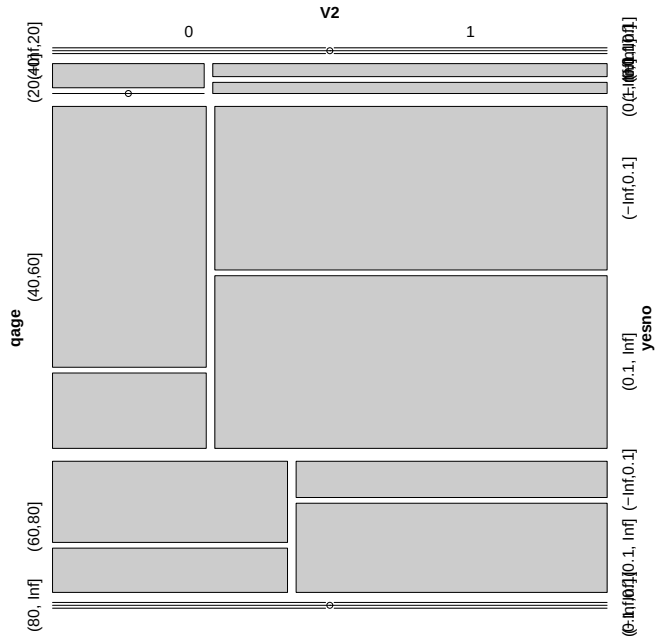

FIGURE 11.11: *This is a mosaic plot of age, gender, and level of disease for the heart dataset from the UC Irvine repository. Notice that the data consists mainly of people aged 40-60. There are very few people younger than 20 or older than 80. A significantly greater percentage of the measurements comes from the gender labelled 1, and in all age groups the percentage of that gender that has the disease level is higher. This suggests that this gender is male. Notice also that the percentage of males with the disease is really quite high (at least 50% in each case). This suggests that either the population is special in some way — perhaps the measurements are collected from people who are feeling sick — or that the criterion used to determine whether an individual is diseased is too sensitive. R code for this plot is on the website.*

transforming data to a uniform blob could be helpful.

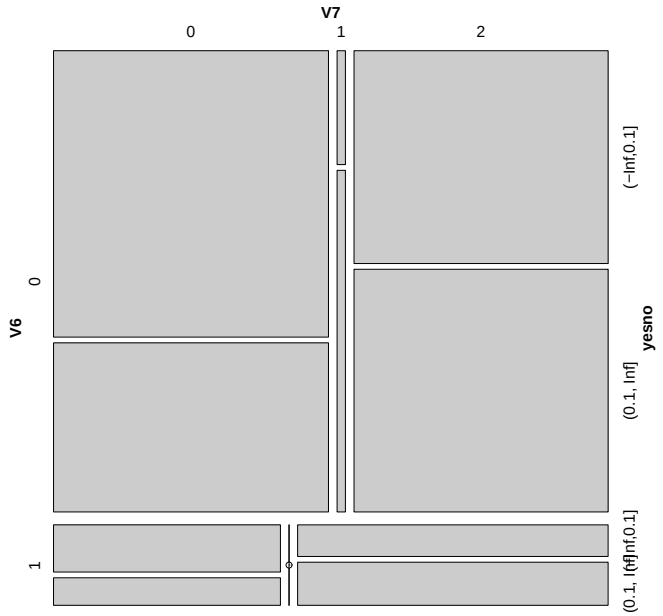

FIGURE 11.12: *This is a mosaic plot of variables six (V6) and seven (V7) and level of disease for the heart dataset from the UC Irvine repository. These variables represent some physiological properties of importance, but I don't know their interpretation. V6 does not seem to be particularly significant, but the population for which V7 has value 1 has a high incidence of disease. R code for this plot is on the website.*

Useful Facts: 11.1 *Whitening a dataset*

For a dataset $\{\mathbf{x}\}$, compute:

- $\mathcal{U}$, the matrix of eigenvectors of $\mathsf{Covmat}(\{\mathbf{x}\})$;
- and $\mathsf{mean}(\{\mathbf{x}\})$.

Now compute $\{\mathbf{n}\}$ using the rule

$$\mathbf{n}_i = \mathcal{U}^T(\mathbf{x}_i - \mathsf{mean}(\{\mathbf{x}\})).$$

Then $\mathsf{mean}(\{\mathbf{n}\}) = \mathbf{0}$ and $\mathsf{Covmat}(\{\mathbf{n}\})$ is diagonal.
Now write Λ for the diagonal matrix of eigenvalues of $\mathsf{Covmat}(\{\mathbf{x}\})$ (so that $\mathsf{Covmat}(\{\mathbf{x}\})\mathcal{U} = \mathcal{U}\Lambda$). Assume that each of the diagonal entries of Λ is greater than zero (otherwise there is a redundant dimension in the data). Write λ_i for the i'th diagonal entry of Λ, and write $\Lambda^{-(1/2)}$ for the diagonal matrix whose i'th diagonal entry is $1/\sqrt{\lambda_i}$. Compute $\{\mathbf{z}\}$ using the rule

$$\mathbf{z}_i = \Lambda^{(-1/2)}\mathcal{U}(\mathbf{x}_i - \mathsf{mean}(\{\mathbf{x}\})).$$

We have that $\mathsf{mean}(\{\mathbf{z}\}) = 0$ and $\mathsf{Covmat}(\{\mathbf{z}\}) = \mathcal{I}$. The dataset $\{\mathbf{z}\}$ is often known as **whitened data**.

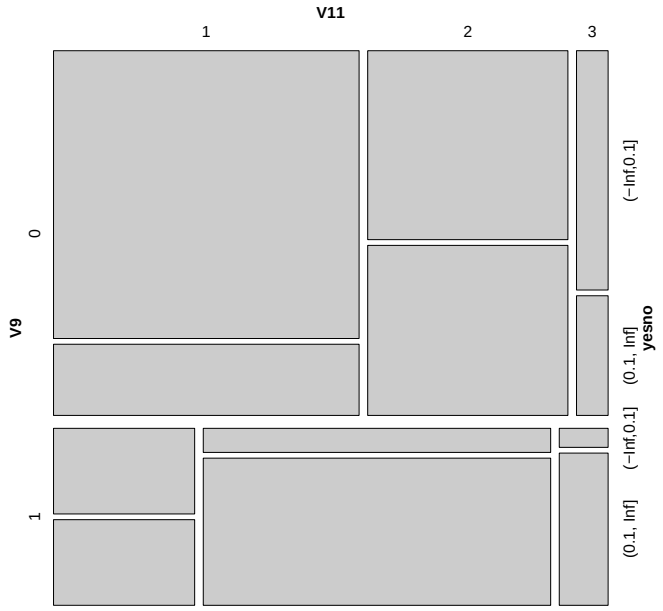

FIGURE 11.13: *This is a mosaic plot of variables nine (V9) and eleven (V11) and level of disease for the heart dataset from the UC Irvine repository. These variables represent some physiological properties of importance, but I don't know their interpretation. V9 at level one is clearly not a good thing. If V9 is at level 0, then V11 at level 1 is also a problem. R code for this plot is on the website.*

It isn't always a good idea to whiten data. In some circumstances, each separate component is meaningful, and in a meaningful set of units. For example, one of the components might be a length using a natural scale and the other might be a time on a natural scale. When this happens, we might be reluctant to transform the data, either because we don't want to add lengths to times or because we want to preserve the scales.

11.3 PRINCIPAL COMPONENTS ANALYSIS

Mostly, when one deals with high dimensional data, it isn't clear which individual components are important. As we have seen with the height weight dataset (for example, in the case of density and weight) some components can be quite strongly correlated. Equivalently, as in Figure 1, the blob is not aligned with the coordinate axes.

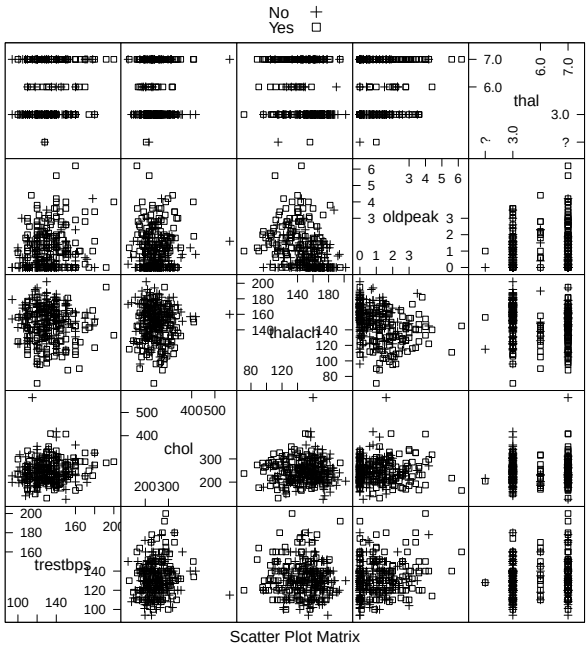

FIGURE 11.14: *This is a scatterplot matrix of five variables for the heart dataset from the UC Irvine repository. The shape of the marker indicates disease or not. The variable names are taken from the description at that URL. "trestbps" is a measurement of resting systolic blood pressure, and "chol" of cholesterol (I don't know which lipid, or in which units). You should notice that "thal" takes only a discrete set of values. Notice also that it appears to be unwise to have very large values of "trestbps", "chol", or "oldpeak" (or smal values of "thalach"), it isn't that easy to distinguish between the different cases. There isn't a clear clustering the way there was in the iris data. R code for this plot is on the website.*

11.3.1 The Blob Coordinate System and Smoothing

We can use the fact that we *could* rotate, translate and scale the blob to define a coordinate system within the blob. The origin of that coordinate system is the mean of the data, and the coordinate axes are given by the eigenvectors of the covariance matrix. These are orthonormal, so they form a set of unit basis vectors at right angles to one another (i.e. a coordinate system). You should think of these as blob coordinates; Figure 11.20 illustrates a set of blob coordinates.

The blob coordinate system is important because, *once we know the blob coordinate system*, we can identify important scales of variation in the data. For example, if you look at Figure 11.20, you can see that this blob is extended much further along one direction than along the other. We can use this information to identify the most significant forms of variation in very high dimensional data. In some directions in the blob coordinate system, the blob will be spread out — ie have large variance — but in others, it might not be.

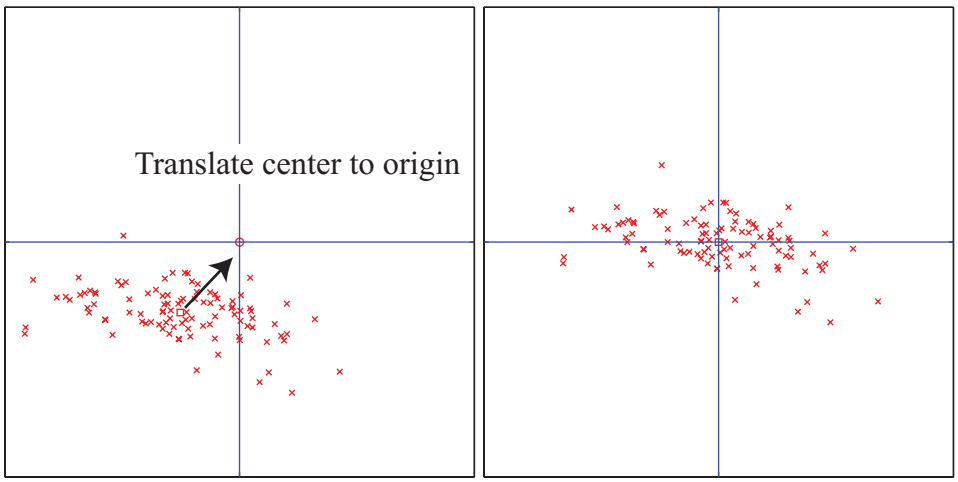

FIGURE 11.15: *On the* **left**, *a "blob" in two dimensions. This is a set of data points that lie somewhat clustered around a single center, given by the mean. I have plotted the mean of these data points with a hollow square (it's easier to see when there is a lot of data). To translate the blob to the origin, we just subtract the mean from each datapoint, yielding the blob on the* **right**.

Equivalently, imagine we choose to represent each data item *in blob coordinates*. Then the mean over the dataset will be zero. Each pair of distinct coordinates will be uncorrelated. Some coordinates — corresponding to directions where the blob is spread out — will have a large range of values. Other coordinates — directions in which the blob is small — will have a small range of values. We could choose to replace these coordinates with zeros, with little significant loss in accuracy. The advantage of doing so is that we would have lower dimensional data to deal with.

However, it isn't particularly natural to work in blob coordinates. Each component of a data item may have a distinct meaning and scale (i.e. feet, pounds, and so on), but this is not preserved in any easy way in blob coordinates. Instead, we should like to (a) compute a lower dimensional representation in blob coordinates then (b) transform that representation into the original coordinate system of the data item. Doing so is a form of **smoothing** — suppressing small, irrelevant variations by exploiting multiple data items.

For example, look at Figure 11.21. Imagine we transform the blob on the left to blob coordinates. The covariance matrix in these coordinates is a 3×3 diagonal matrix. One of the values on the diagonal is large, because the blob is extended on one direction; but the other two are small. This means that, in blob coordinates, the data varies significantly in one direction, but very little in the other two directions.

Now imagine we project the data points onto the high-variation direction; equivalently, we set the other two directions to zero for each data point. Each of the new data points is very close to the corresponding old data point, because by setting the small directions to zero we haven't moved the point very much.

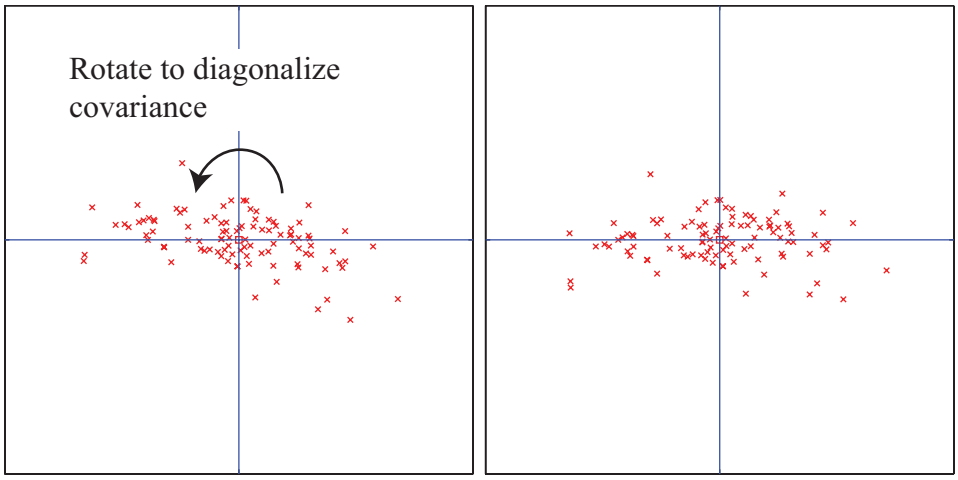

FIGURE 11.16: *On the* **left**, *the translated blob of figure 11.15. This blob lies somewhat diagonally, because the vertical and horizontal components are correlated. On the* **right**, *that blob of data rotated so that there is no correlation between these components. We can now describe the blob by the vertical and horizontal variances alone, as long as we do so in the new coordinate system. In this coordinate system, the vertical variance is significantly larger than the horizontal variance — the blob is short and wide.*

In blob coordinates, the covariance matrix of this new dataset has changed very little. It is again a 3×3 diagonal matrix, but now two of the diagonal values are zero, because there isn't any variance in those directions. The third value is large, because the blob is extended in that direction. We take the new dataset, and rotate and translate it into the original coordinate system. Each point must lie close to the corresponding point in the original dataset. However, the new dataset lies along a straight line (because it lay on a straight line in the blob coordinates). This process gets us the blob on the right in Figure 11.21. This blob is a smoothed version of the original blob.

Smoothing works because when two data items are strongly correlated, the value of one is a good guide to the value of the other. This principle works for more than two data items. Section 1 describes an example where the data items have dimension 101, but all values are extremely tightly correlated. In a case like this, there may be very few dimensions in blob coordinates that have any significant variation (3-6 for this case, depending on some details of what one believes is a small number, and so on). The components are so strongly correlated in this case that the 101-dimensional blob really looks like a slightly thickened 3 (or slightly more) dimensional blob that has been inserted into a 101-dimensional space (Figure 11.21). If we project the 101-dimensional data onto that structure in the original, 101-dimensional space, we may get much better estimates of the components of each data item than the original measuring device can supply. This occurs because each component is now estimated using correlations between all the measurements.

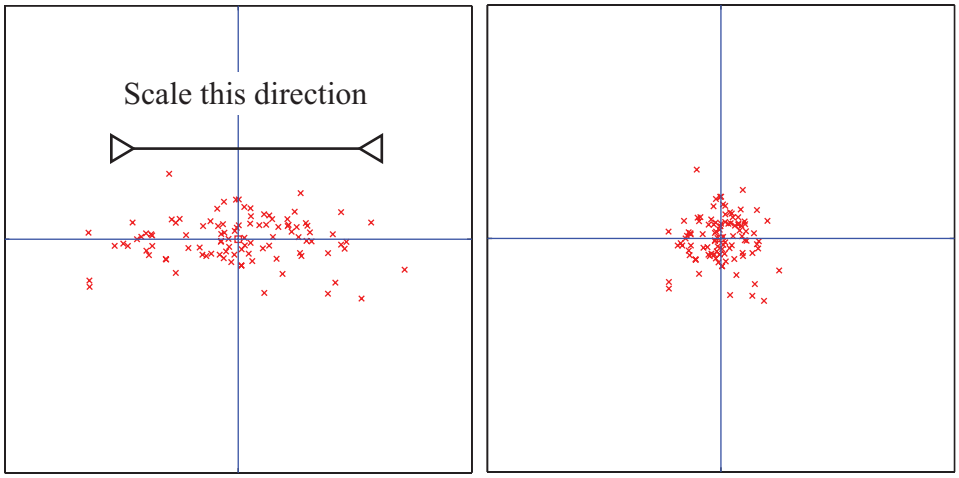

FIGURE 11.17: *On the* **left**, *the translated and rotated blob of figure 11.16. This blob is stretched — one direction has more variance than another. Because all covariances are zero, it is easy to scale the blob so that all variances are one (the blob on the* **right**). *You can think of this as a standard blob. All blobs can be reduced to a standard blob, by relatively straightforward linear algebra.*

11.3.2 The Low-Dimensional Representation of a Blob

We wish to construct an r dimensional representation of a blob, where we have chosen r in advance. First, we compute $\{\mathbf{v}\}$ by translating the blob so its mean is at the origin, so that $\mathbf{v}_i = \mathbf{x}_i - \mathsf{mean}\,(\{\mathbf{x}\})$. Now write $\mathcal{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N]$. The covariance matrix of $\{\mathbf{v}\}$ is then

$$\mathsf{Covmat}\,(\{\mathbf{v}\}) = \frac{1}{N}\mathcal{V}\mathcal{V}^T = \mathsf{Covmat}\,(\{\mathbf{x}\}).$$

Now write Λ for the diagonal matrix of eigenvalues of $\mathsf{Covmat}\,(\{\mathbf{x}\})$ and $\mathcal{U}$ for the matrix of eigenvectors, so that $\mathsf{Covmat}\,(\{\mathbf{x}\})\mathcal{U} = \mathcal{U}\Lambda$. We assume that the elements of Λ are sorted in decreasing order along the diagonal. The covariance matrix for the dataset transformed into blob coordinates will be Λ. Notice that

$$\begin{aligned}\Lambda &= \mathcal{U}^T \mathsf{Covmat}\,(\{\mathbf{x}\})\mathcal{U} \\ &= \mathcal{U}^T\mathcal{V}\mathcal{V}^T\mathcal{U} \\ &= (\mathcal{U}^T\mathcal{V})(\mathcal{U}^T\mathcal{V})^T.\end{aligned}$$

This means we can interpret $(\mathcal{U}^T\mathcal{V})$ as a new dataset $\{\mathbf{b}\}$. This is our data, rotated into blob coordinates.

Now write Π_r for the $d \times d$ matrix

$$\begin{bmatrix} \mathcal{I}_r & 0 \\ 0 & 0 \end{bmatrix}$$

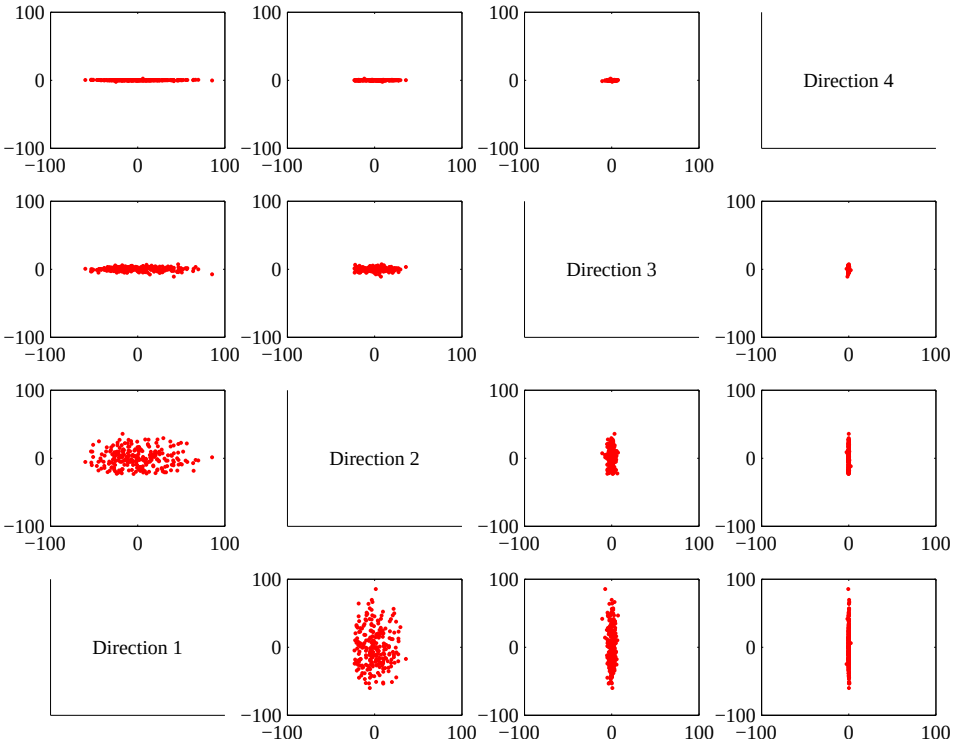

FIGURE 11.18: *A panel plot of the bodyfat dataset of figure 11.4, now rotated so that the covariance between all pairs of distinct dimensions is zero. Now we do not know names for the directions — they're linear combinations of the original variables. Each scatterplot is on the same set of axes, so you can see that the dataset extends more in some directions than in others.*

which projects a d dimensional vector onto its first r components, and replaces the others with zeros. Then we have that

$$\Lambda_r = \Pi_r \Lambda \Pi_r^T$$

is the covariance matrix for the reduced dimensional data in blob coordinates. Notice that Λ_r keeps the r largest eigenvalues on the diagonal of Λ, and replaces all others with zero.

We have

$$\begin{aligned} \Lambda_r &= \Pi_r \Lambda \Pi_r^T \\ &= \Pi_r \mathcal{U}^T \mathsf{Covmat}(\{\mathbf{x}\}) \mathcal{U} \Pi_r^T \\ &= (\Pi_r \mathcal{U}^T \mathcal{V})(\mathcal{V}^T \mathcal{U} \Pi_r^T) \\ &= \mathcal{P}\mathcal{P}^T \end{aligned}$$

where $\mathcal{P} = (\Pi_r \mathcal{U}^T \mathcal{V})$. This represents our data, rotated into blob coordinates,

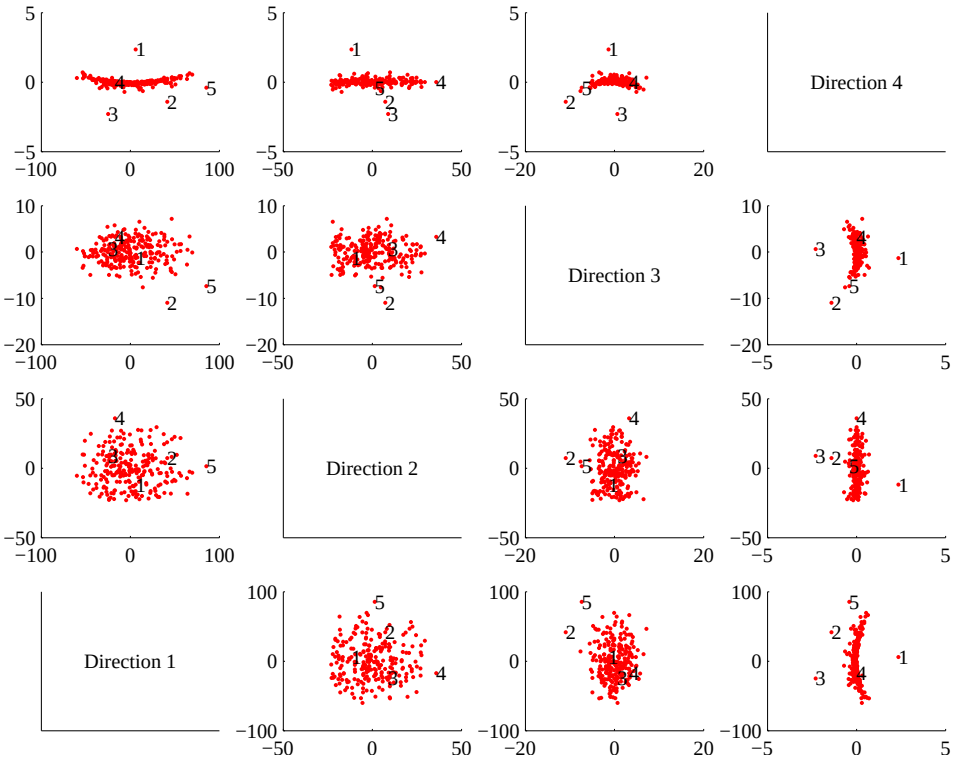

FIGURE 11.19: *A panel plot of the bodyfat dataset of figure 11.4, now rotated so that the covariance between all pairs of distinct dimensions is zero. Now we do not know names for the directions — they're linear combinations of the original variables. I have scaled the axes so you can see details; notice that the blob is a little curved, and there are several data points that seem to lie some way away from the blob, which I have numbered.*

and then projected down to r dimensions, with remaining terms replaced by zeros. Write $\{\mathbf{b}_r\}$ for this new dataset.

Occasionally, we need to refer to this representation, and we give it a special name. Write

$$\mathsf{pcaproj}\left(\mathbf{x}_i, r, \{\mathbf{x}\}\right) = \Pi_r \mathcal{U}^T (\mathbf{x}_i - \mathsf{mean}\left(\{\mathbf{x}\}\right))$$

where the notation seeks to explicitly keep track of the fact that the low dimensional representation of a particular data item *depends on the whole dataset* (because you have to be able to compute the mean, and the eigenvectors of the covariance). Notice that $\mathsf{pcaproj}\left(\mathbf{x}_i, r, \{\mathbf{x}\}\right)$ is a representation of the dataset with important properties:

- The representation is r-dimensional (i.e. the last $d - r$ components are zero).
- Each pair of distinct components of $\{\mathsf{pcaproj}\left(\mathbf{x}_i, r, \{\mathbf{x}\}\right)\}$ has zero covariance.

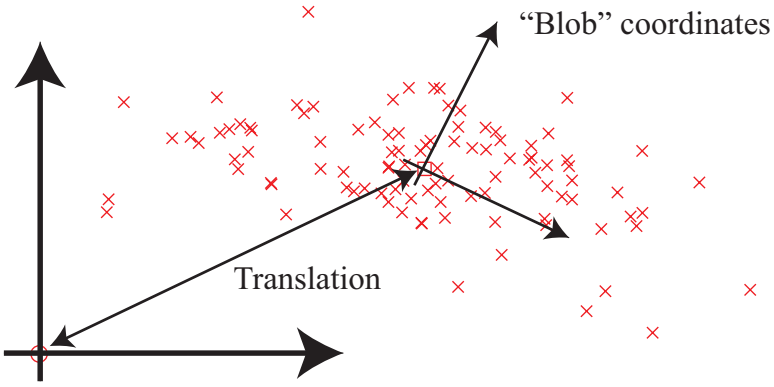

FIGURE 11.20: *A 2D blob, with its natural blob coordinate system. The origin of this coordinate system is at the mean of the data. The coordinate axes are (a) at right angles to one another and (b) are directions that have no covariance.*

- The first component of $\{\mathsf{pcaproj}\,(\mathbf{x}_i, r, \{\mathbf{x}\})\}$ has largest variance; the second component has second largest variance; and so on.

11.3.3 Smoothing Data with a Low-Dimensional Representation

We would now like to construct a low dimensional representation of the blob, in the original coordinates. We do so by rotating the low-dimensional representation back to the original coordinate system, then adding back the mean to translate the origin back to where it started. We can write this as

$$\begin{aligned}\mathsf{pcasmooth}\,(\mathbf{x}_i, r, \{\mathbf{x}\}) &= \mathcal{U}\Pi_r^T\left(\Pi_r\mathcal{U}^T(\mathbf{x}_i - \mathsf{mean}\,(\{\mathbf{x}\}))\right) + \mathsf{mean}\,(\{\mathbf{x}\}) \\ &= \mathcal{U}\Pi_r^T\,\mathsf{pcaproj}\,(\mathbf{x}_i, r, \{\mathbf{x}\}) + \mathsf{mean}\,(\{\mathbf{x}\})\end{aligned}$$

we have a new representation of the i'th data item *in the original space* (Figure 1). Now consider the dataset obtained by smoothing each of our data items. We write this dataset as $\{\mathsf{pcasmooth}\,(\mathbf{x}_i, r, \{\mathbf{x}\})\}$.

You should think of $\{\mathsf{pcasmooth}\,(\mathbf{x}_i, r, \{\mathbf{x}\})\}$ as a smoothed version of the original data. One way to think of this process is that we have chosen a low-dimensional basis that represents the main variance in the data rather well. It is quite usual to think of a data item as being given by a the mean plus a weighted sum of these basis elements. In this view, the first weight has larger variance than the second, and so on. By construction, this dataset lies in an r dimensional affine subspace of the original space. We constructed this r-dimensional space to preserve the largest variance directions of the data. Each column of this matrix is known as a **principal component**. In particular, we have constructed this dataset so that

- $\mathsf{mean}\,(\{\mathsf{pcasmooth}\,(\mathbf{x}_i, r, \{\mathbf{x}\})\}) = \mathsf{mean}\,(\{\mathbf{x}\})$;
- $\mathsf{Covmat}\,(\{\mathsf{pcasmooth}\,(\mathbf{x}_i, r, \{\mathbf{x}\})\})$ has rank r;
- $\mathsf{Covmat}\,(\{\mathsf{pcasmooth}\,(\mathbf{x}_i, r, \{\mathbf{x}\})\})$ is the best approximation of $\mathsf{Covmat}\,(\{\mathbf{x}\})$ with rank r.

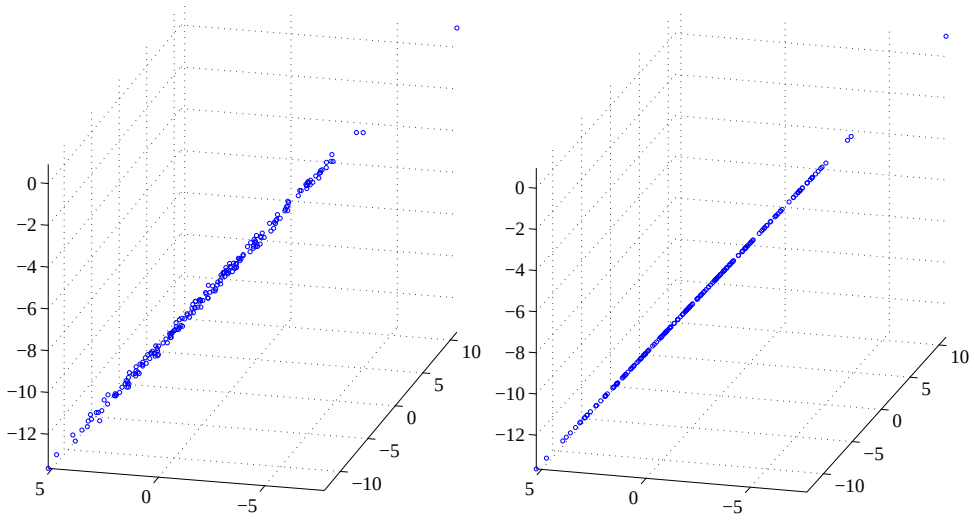

FIGURE 11.21: *On the* **left**, *a blob of 3D data that has very low variance in two directions in the blob coordinates. As a result, all the data points are very close to a 1D blob. Experience shows that this is a common phenomenon. Although there might be many components in the data items, all data points are very close to a much lower dimensional object in the high dimensional space. When this is the case, we could obtain a lower dimensional representation of the data by working in blob coordinates, or we could smooth the data (as on the* **right**), *by projecting each data point onto the lower dimensional space.*

Figure 11.23 gives a visualization of the smoothing process. By comparing figures 11.18 and 11.24, you can see that a real dataset can lose several dimensions without much significant going wrong. As we shall see in the examples, some datasets can lose many dimensions without anything bad happening.

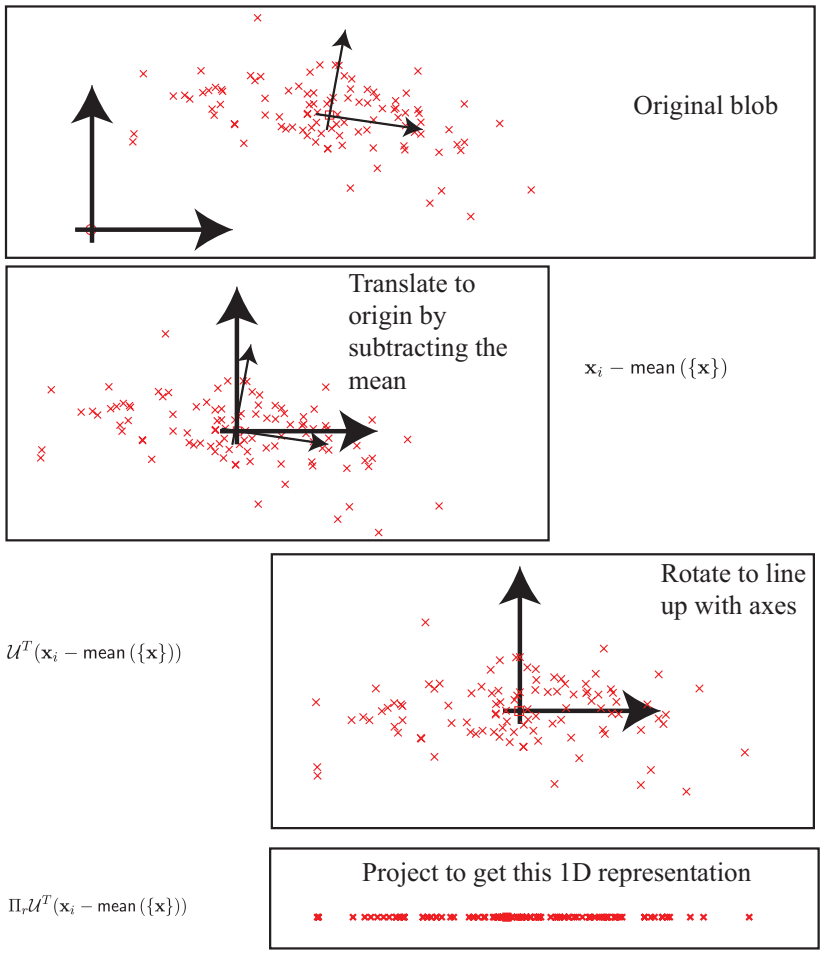

FIGURE 11.22: *Computing a low dimensional representation for principal components analysis.*

Procedure: 11.1 *Principal Components Analysis*

Assume we have a general data set $\mathbf{x}_i$, consisting of N d-dimensional vectors. Now write $\Sigma = \text{Covmat}(\{\mathbf{x}\})$ for the covariance matrix.
Form $\mathcal{U}$, Λ, such that $\Sigma\mathcal{U} = \mathcal{U}\Lambda$ (these are the eigenvectors and eigenvalues of Σ). Ensure that the entries of Λ are sorted in decreasing order. Choose r, the number of dimensions you wish to represent. Typically, we do this by plotting the eigenvalues and looking for a "knee" (Figure **??**). It is quite usual to do this by hand.
Constructing a low-dimensional representation: Form $\mathcal{U}_r$, a matrix consisting of the first r columns of $\mathcal{U}$. Now compute $\{\text{pcaproj}(\mathbf{x}_i, r, \{\mathbf{x}\})\} = \{(\Pi_r\mathcal{U}^T(\mathbf{x}_i - \text{mean}(\{\mathbf{x}\})))\}$. This is a set of data vectors which are r dimensional, and where each component is independent of each other component (i.e. the covariances of distinct components are zero).
Smoothing the data: Form $\{\text{pcasmooth}(\mathbf{x}_i, r, \{\mathbf{x}\})\} = \{(\mathcal{U}_r\text{pcaproj}(\mathbf{x}_i, r, \{\mathbf{x}\}) + \text{mean}(\{\mathbf{x}\}))\}$. These are d dimensional vectors that lie in a r-dimensional subspace of d-dimensional space. The "missing dimensions" have the lowest variance, and are independent.

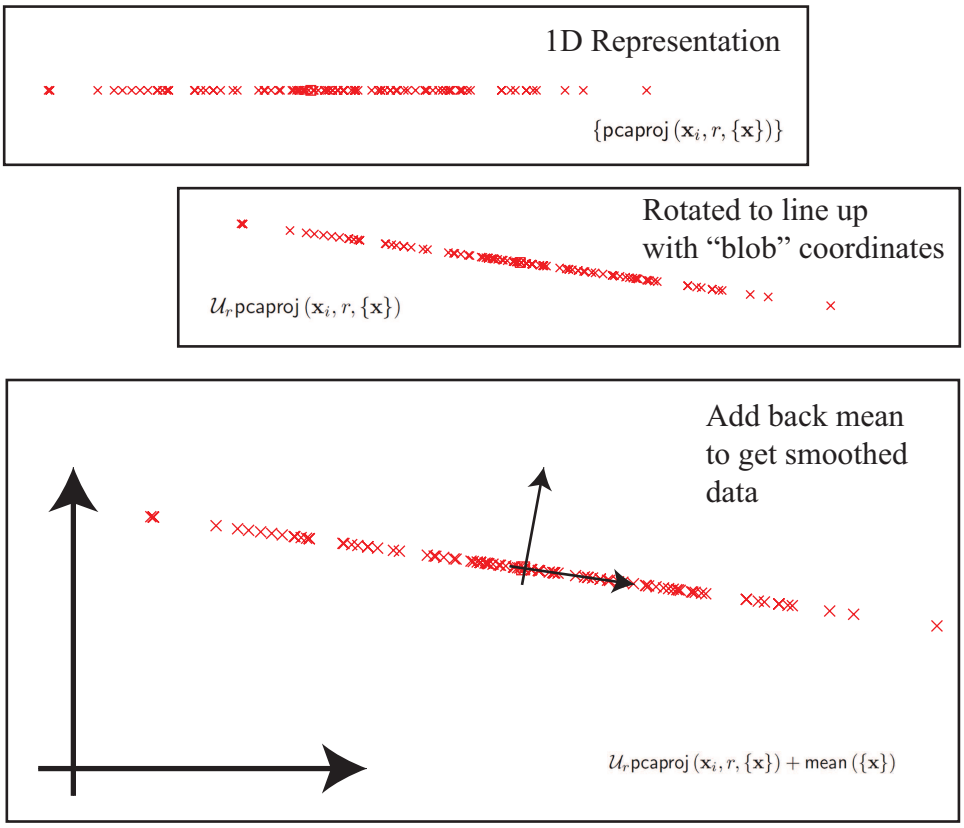

FIGURE 11.23: *Smoothing data with principal components analysis.*

11.3.4 The Error of the Low-Dimensional Representation

We took a dataset, $\{\mathbf{x}\}$, and constructed a d-dimensional dataset $\{\mathbf{b}\}$ in blob coordinates. We did so by translating, then rotating, the data, so no information was lost; we could reconstruct our original dataset by rotating, then translating $\{\mathbf{b}\}$. But in blob coordinates we projected each data item down to the first r components to get an r-dimensional dataset $\{\mathbf{b}_r\}$. We then reconstructed a smoothed dataset by rotating, then translating, $\{\mathbf{b}_r\}$. Information has been lost here, but how much?

The answer is easy to get if you recall that rotations and translations do not change lengths. This means that

$$\| \mathbf{x}_i - \text{pcasmooth}\,(\mathbf{x}_i, r, \{\mathbf{x}\}) \|^2 = \| \mathbf{b}_i - \mathbf{b}_{r,i} \|^2.$$

This expression is easy to evaluate, because $\mathbf{b}_i$ and $\mathbf{b}_{r,i}$ agree in their first r com-

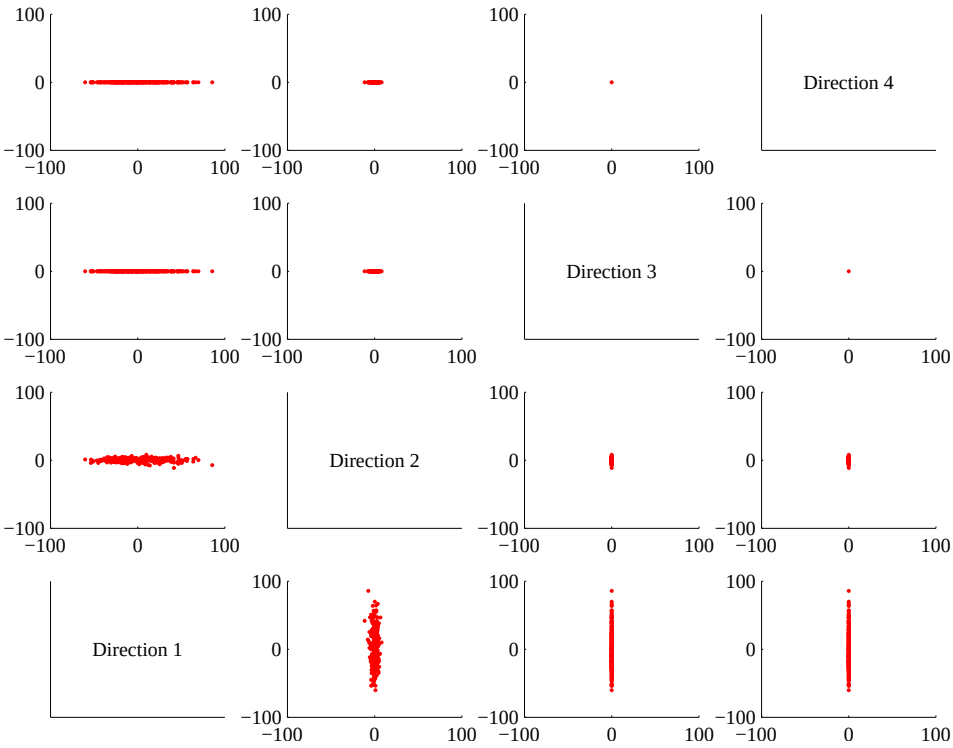

FIGURE 11.24: *A panel plot of the bodyfat dataset of figure 11.4, with the dimension reduced to two using principal components analysis. Compare this figure to figure 11.18, which is on the same set of axes. You can see that the blob has been squashed in direction 3 and direction 4. But not much has really happened, because there wasn't very much variation in those directions in the first place.*

ponents. The remaining $d - r$ components of $\mathbf{b}_{r,i}$ are zero. So we can write

$$\| \mathbf{x}_i - \mathsf{pcasmooth}\left(\mathbf{x}_i, r, \{\mathbf{x}\}\right) \|^2 = \sum_{u=r+1}^{d} (\mathbf{b}_i^{(u)})^2.$$

Now a natural measure of error is the average over the dataset of this term. We have that

$$\frac{1}{N} \sum_{u=r+1}^{d} (\mathbf{b}_i^{(u)})^2 = \sum_{u=r+1}^{d} \mathsf{var}\left(\left\{\mathbf{b}^{(u)}\right\}\right)$$

which is easy to evaluate, because we know these variances — they are the values of the $d-r$ eigenvalues that we decided to ignore. So the mean error can be written as

$$\mathbf{1}^T(\Lambda - \Lambda_r)\mathbf{1}.$$

Now we could choose r by identifying how much error we can tolerate. More usual

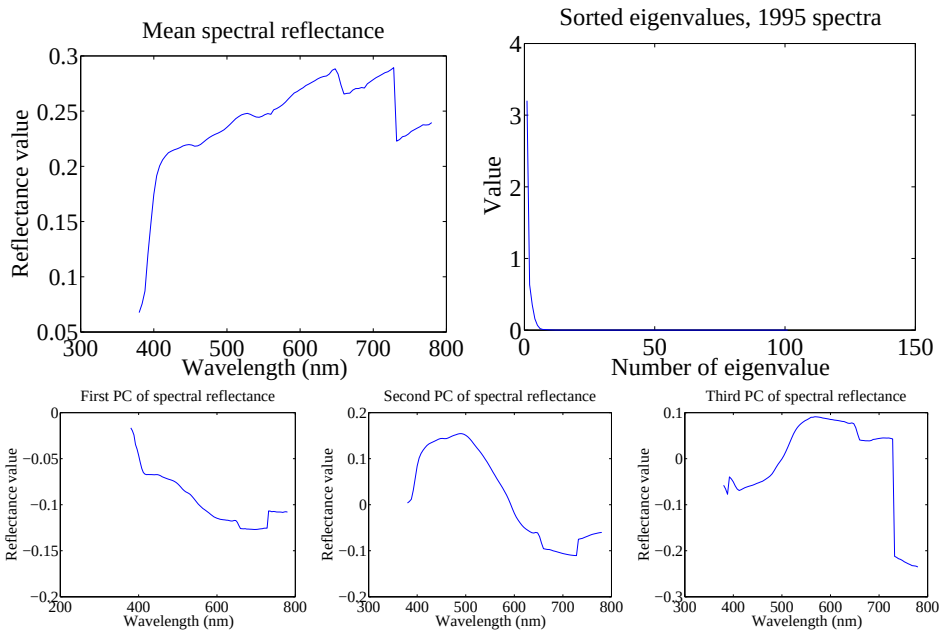

FIGURE 11.25: *On the* **top left**, *the mean spectral reflectance of a dataset of 1995 spectral reflectances, collected by Kobus Barnard (at* `http://www.cs.sfu.ca/~colour/data/`*). On the* **top right**, *eigenvalues of the covariance matrix of spectral reflectance data, from a dataset of 1995 spectral reflectances, collected by Kobus Barnard (at* `http://www.cs.sfu.ca/~colour/data/`*). Notice how the first few eigenvalues are large, but most are very small; this suggests that a good representation using few principal components is available. The* **bottom row** *shows the first three principal components. A linear combination of these, with appropriate weights, added to the mean of figure* **??**, *gives a good representation of the dataset.*

is to plot the eigenvalues of the covariance matrix, and look for a "knee", like that in Figure 1. You can see that the sum of remaining eigenvalues is small.

11.3.5 Example: Representing Spectral Reflectances

Diffuse surfaces reflect light uniformly in all directions. Examples of diffuse surfaces include matte paint, many styles of cloth, many rough materials (bark, cement, stone, etc.). One way to tell a diffuse surface is that it does not look brighter (or darker) when you look at it along different directions. Diffuse surfaces can be colored, because the surface reflects different fractions of the light falling on it at different wavelengths. This effect can be represented by measuring the spectral reflectance of a surface, which is the fraction of light the surface reflects as a function of wavelength. This is usually measured in the visual range of wavelengths (about 380nm to about 770 nm). Typical measurements are every few nm, depending on the measurement device. I obtained data for 1995 different surfaces from `http:`

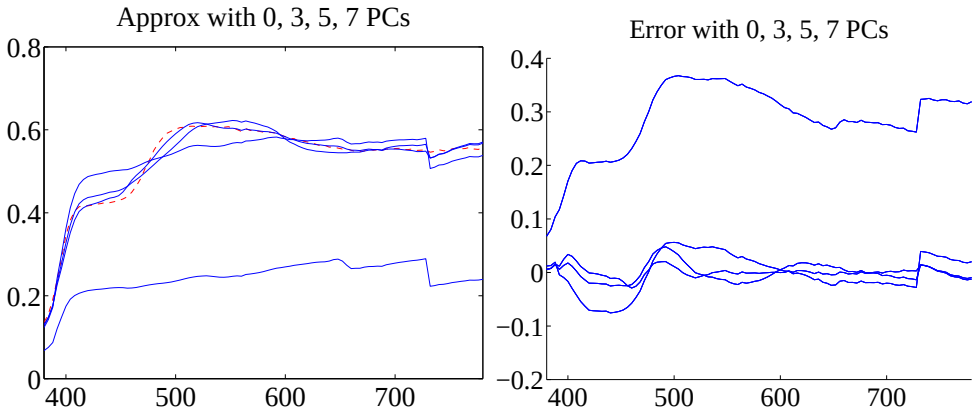

FIGURE 11.26: *On the* **left**, *a spectral reflectance curve (dashed) and approximations using the mean, the mean and 3 principal components, the mean and 5 principal components, and the mean and 7 principal components. Notice the mean is a relatively poor approximation, but as the number of principal components goes up, the error falls rather quickly. On the* **right** *is the error for these approximations. Figure plotted from a dataset of 1995 spectral reflectances, collected by Kobus Barnard (at* `http://www.cs.sfu.ca/~colour/data/`*).*

`//www.cs.sfu.ca/~colour/data/` (there are a variety of great datasets here, from Kobus Barnard).

Each spectrum has 101 measurements, which are spaced 4nm apart. This represents surface properties to far greater precision than is really useful. Physical properties of surfaces suggest that the reflectance can't change too fast from wavelength to wavelength. It turns out that very few principal components are sufficient to describe almost any spectral reflectance function. Figure 11.25 shows the mean spectral reflectance of this dataset, and Figure 11.25 shows the eigenvalues of the covariance matrix.

This is tremendously useful in practice. One should think of a spectral reflectance as a function, usually written $\rho(\lambda)$. What the principal components analysis tells us is that we can represent this function rather accurately on a (really small) finite dimensional basis. This basis is shown in figure 11.25. This means that there is a mean function $r(\lambda)$ and k functions $\phi_m(\lambda)$ such that, for any $\rho(\lambda)$,

$$\rho(\lambda) = r(\lambda) + \sum_{i=1}^{k} c_i \phi_i(\lambda) + e(\lambda)$$

where $e(\lambda)$ is the error of the representation, which we know is small (because it consists of all the other principal components, which have tiny variance). In the case of spectral reflectances, using a value of k around 3-5 works fine for most applications (Figure 11.26). This is useful, because when we want to predict what a particular object will look like under a particular light, we don't need to use a detailed spectral reflectance model; instead, it's enough to know the c_i for that

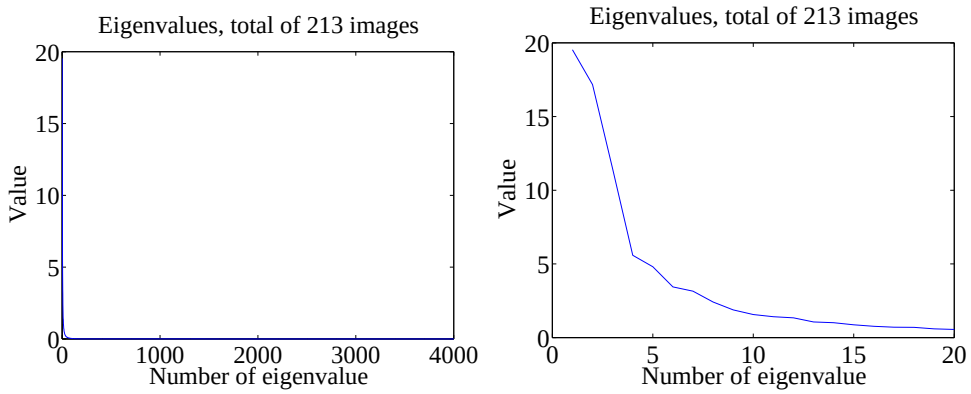

FIGURE 11.27: *On the* **left**,*the eigenvalues of the covariance of the Japanese facial expression dataset; there are 4096, so it's hard to see the curve (which is packed to the left). On the* **right**, *a zoomed version of the curve, showing how quickly the values of the eigenvalues get small.*

object. This comes in useful in a variety of rendering applications in computer graphics. It is also the key step in an important computer vision problem, called **color constancy**. In this problem, we see a picture of a world of colored objects under unknown colored lights, and must determine what color the objects are. Modern color constancy systems are quite accurate, even though the problem sounds underconstrained. This is because they are able to exploit the fact that relatively few c_i are enough to accurately describe a surface reflectance.

11.3.6 Example: Representing Faces with Principal Components

An image is usually represented as an array of values. We will consider intensity images, so there is a single intensity value in each cell. You can turn the image into a vector by rearranging it, for example stacking the columns onto one another (use `reshape` in Matlab). This means you can take the principal components of a set of images. Doing so was something of a fashionable pastime in computer vision for a while, though there are some reasons that this is not a great representation of pictures. However, the representation yields pictures that can give great intuition into a dataset.

Figure **??** shows the mean of a set of face images encoding facial expressions of Japanese women (available at `http://www.kasrl.org/jaffe.html`; there are tons of face datasets at `http://www.face-rec.org/databases/`). I reduced the images to 64x64, which gives a 4096 dimensional vector. The eigenvalues of the covariance of this dataset are shown in figure 11.27; there are 4096 of them, so it's hard to see a trend, but the zoomed figure suggests that the first couple of hundred contain most of the variance. Once we have constructed the principal components, they can be rearranged into images; these images are shown in figure 11.28. Principal components give quite good approximations to real images (figure 11.29).

The principal components sketch out the main kinds of variation in facial

FIGURE 11.28: *The mean and first 16 principal components of the Japanese facial expression dataset.*

expression. Notice how the mean face in Figure 11.28 looks like a relaxed face, but with fuzzy boundaries. This is because the faces can't be precisely aligned, because each face has a slightly different shape. The way to interpret the components is to remember one adjusts the mean towards a data point by adding (or subtracting) some scale times the component. So the first few principal components have to do with the shape of the haircut; by the fourth, we are dealing with taller/shorter faces; then several components have to do with the height of the eyebrows, the shape of the chin, and the position of the mouth; and so on. These are all images of women who are not wearing spectacles. In face pictures taken from a wider set of models, moustaches, beards and spectacles all typically appear in the first couple of dozen principal components.

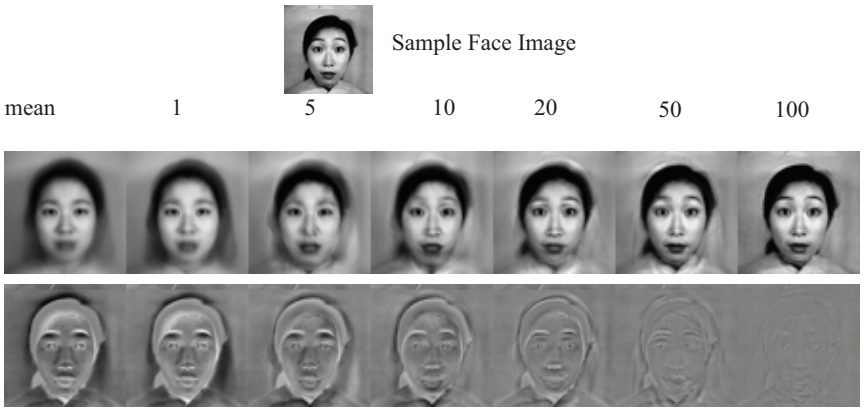

FIGURE 11.29: *Approximating a face image by the mean and some principal components; notice how good the approximation becomes with relatively few components.*

11.4 MULTI-DIMENSIONAL SCALING

One way to get insight into a dataset is to plot it. But choosing what to plot for a high dimensional dataset could be difficult. Assume we must plot the dataset in two dimensions (by far the most common choice). We wish to build a scatter plot in two dimensions — but where should we plot each data point? One natural requirement is that the points be laid out in two dimensions in a way that reflects how they sit in many dimensions. In particular, we would like points that are far apart in the high dimensional space to be far apart in the plot, and points that are close in the high dimensional space to be close in the plot.

11.4.1 Principal Coordinate Analysis

We will plot the high dimensional point $\mathbf{x}_i$ at $\mathbf{v}_i$, which is a two-dimensional vector. Now the squared distance between points i and j in the high dimensional space is

$$D_{ij}^{(2)}(\mathbf{x}) = (\mathbf{x}_i - \mathbf{x}_j)^T(\mathbf{x}_i - \mathbf{x}_j)$$

(where the superscript is to remind you that this is a squared distance). We could build an $N \times N$ matrix of squared distances, which we write $\mathcal{D}^{(2)}(\mathbf{x})$. The i, j'th entry in this matrix is $D_{ij}^{(2)}(\mathbf{x})$, and the $\mathbf{x}$ argument means that the distances are between points in the high-dimensional space. Now we could choose the $\mathbf{v}_i$ to make

$$\sum_{ij} \left(D_i^{(2)} j(\mathbf{x}) - D_{ij}^{(2)}(\mathbf{v}) \right)^2$$

as small as possible. Doing so should mean that points that are far apart in the high dimensional space are far apart in the plot, and that points that are close in the high dimensional space are close in the plot.

In its current form, the expression is difficult to deal with, but we can refine it. Because translation does not change the distances between points, it cannot

change either of the $\mathcal{D}^{(2)}$ matrices. So it is enough to solve the case when the mean of the points $\mathbf{x}_i$ is zero. We can assume that $\frac{1}{N}\sum_i \mathbf{x}_i = \mathbf{0}$. Now write $\mathbf{1}$ for the n-dimensional vector containing all ones, and $\mathcal{I}$ for the identity matrix. Notice that

$$D_{ij}^{(2)} = (\mathbf{x}_i - \mathbf{x}_j)^T(\mathbf{x}_i - \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_i - 2\mathbf{x}_i \cdot \mathbf{x}_j + \mathbf{x}_j \cdot \mathbf{x}_j.$$

Now write

$$\mathcal{A} = \left[\mathcal{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^T\right].$$

Using this expression, you can show that the matrix $\mathcal{M}$, defined below,

$$\mathcal{M}(\mathbf{x}) = -\frac{1}{2}\mathcal{A}\mathcal{D}^{(2)}(\mathbf{x})\mathcal{A}^T$$

has i, jth entry $\mathbf{x}_i \cdot \mathbf{x}_j$ (exercises). I now argue that, to make $\mathcal{D}^{(2)}(\mathbf{v})$ is close to $\mathcal{D}^{(2)}(\mathbf{x})$, it is enough to make $\mathcal{M}(\mathbf{v})$ close to $\mathcal{M}(\mathbf{x})$. Proving this will take us out of our way unnecessarily, so I omit a proof.

We can choose a set of $\mathbf{v}_i$ that makes $\mathcal{D}^{(2)}(\mathbf{v})$ close to $\mathcal{D}^{(2)}(\mathbf{x})$ quite easily, using the method of the previous section. Take the dataset of N d-dimensional column vectors $\mathbf{x}_i$, and form a matrix $\mathcal{X}$ by stacking the vectors, so

$$\mathcal{X} = \left[\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\right].$$

In this notation, we have

$$\mathcal{M}(\mathbf{x}) = \mathcal{X}^T\mathcal{X}.$$

This matrix is symmetric, and it is positive semidefinite. It can't be positive definite, because the data is zero mean, so $\mathcal{M}(\mathbf{x})\mathbf{1} = 0$. The $\mathcal{M}(\mathbf{v})$ we seek must (a) be as close as possible to $\mathcal{M}(\mathbf{x})$ and (b) have rank 2. It must have rank 2 because there must be some $\mathcal{V}$ which is $2 \times N$ so that $\mathcal{M}(\mathbf{v}) = \mathcal{V}^T\mathcal{V}$. The columns of this $\mathcal{V}$ are our $\mathbf{v}_i$.

We can use the method of section 18.1.1 to construct $\mathcal{M}(\mathbf{v})$ and $\mathcal{V}$. As usual, we write $\mathcal{U}$ for the matrix of eigenvectors of $\mathcal{M}(\mathbf{x})$, Λ for the diagonal matrix of eigenvalues sorted in descending order, Λ_2 for the 2×2 upper left hand block of Λ, and $\Lambda_2^{(1/2)}$ for the matrix of positive square roots of the eigenvalues. Then our methods yield

$$\mathcal{M}(\mathbf{v}) = \mathcal{U}_2\Lambda_2^{(1/2)}\Lambda_2^{(1/2)}\mathcal{U}_2^T$$

and

$$\mathcal{V} = \Lambda_2^{(1/2)}\mathcal{U}_2^T$$

and we can plot these $\mathbf{v}_i$ (example in section 1). This method for constructing a plot is known as **principal coordinate analysis**.

This plot might not be perfect, because reducing the dimension of the data points should cause some distortions. In many cases, the distortions are tolerable. In other cases, we might need to use a more sophisticated scoring system that penalizes some kinds of distortion more strongly than others. There are many ways to do this; the general problem is known as **multidimensional scaling**.

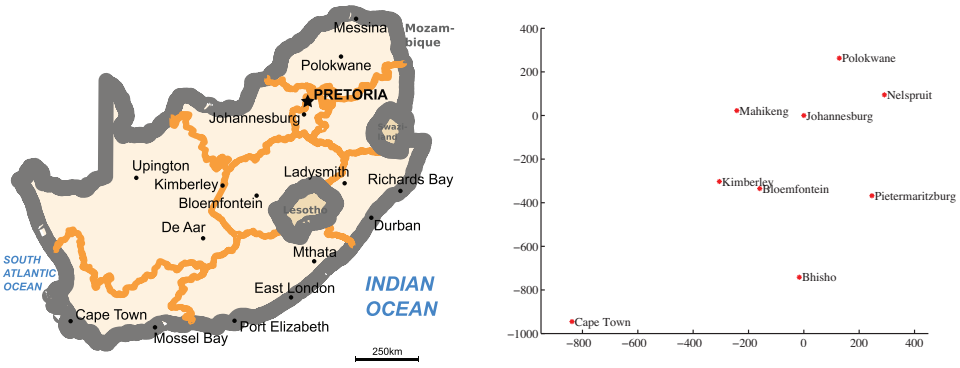

FIGURE 11.30: *On the* **left**, *a public domain map of South Africa, obtained from* `http://commons.wikimedia.org/wiki/File:Map_of_South_Africa.svg`, *and edited to remove surrounding countries. On the* **right**, *the locations of the cities inferred by multidimensional scaling, rotated, translated and scaled to allow a comparison to the map by eye. The map doesn't have all the provincial capitals on it, but it's easy to see that MDS has placed the ones that are there in the right places (use a piece of ruled tracing paper to check).*

Procedure: 11.2 *Principal Coordinate Analysis*

Assume we have a matrix $D^{(2)}$ consisting of the squared differences between each pair of N points. We do not need to know the points. We wish to compute a set of points in r dimensions, such that the distances between these points are as similar as possible to the distances in $D^{(2)}$. Form $\mathcal{A} = \left[\mathcal{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^T\right]$. Form $\mathcal{W} = \frac{1}{2}\mathcal{A}\mathcal{D}^{(2)}\mathcal{A}^T$. Form $\mathcal{U}$, Λ, such that $\mathcal{W}\mathcal{U} = \mathcal{U}\Lambda$ (these are the eigenvectors and eigenvalues of $\mathcal{W}$). Ensure that the entries of Λ are sorted in decreasing order. Choose r, the number of dimensions you wish to represent. Form Λ_r, the top left $r \times r$ block of Λ. Form $\Lambda_r^{(1/2)}$, whose entries are the positive square roots of Λ_r. Form $\mathcal{U}_r$, the matrix consisting of the first r columns of $\mathcal{U}$. Then $\mathcal{V} = \Lambda_2^{(1/2)}\mathcal{U}_2^T = [\mathbf{v}_1, \ldots, \mathbf{v}_N]$ is the set of points to plot.

11.4.2 Example: Mapping with Multidimensional Scaling

Multidimensional scaling gets positions (the $\mathcal{V}$ of section 11.4.1) from distances (the $\mathcal{D}^{(2)}(\mathbf{x})$ of section 11.4.1). This means we can use the method to build maps from distances alone. I collected distance information from the web (I used `http://www.distancefromto.net`, but a google search on "city distances" yields a wide range of possible sources), then apply multidimensional scaling. Table **??** shows distances between the South African provincial capitals, in kilometers, rounded to

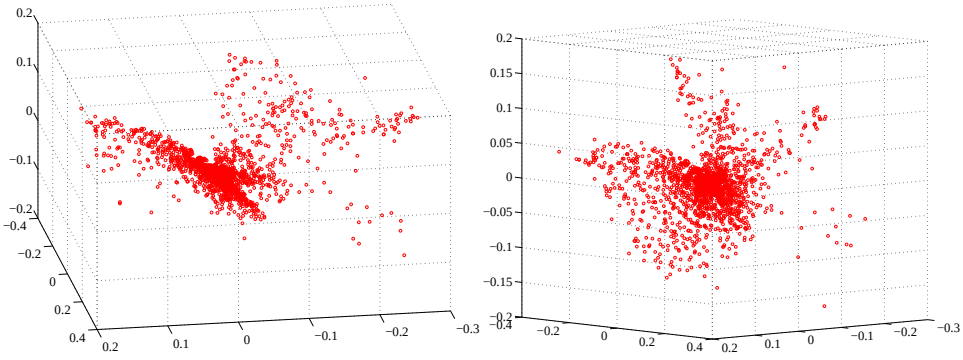

FIGURE 11.31: *Two views of the spectral data of section 11.3.5, plotted as a scatter plot by applying principal coordinate analysis to obtain a 3D set of points. Notice that the data spreads out in 3D, but seems to lie on some structure; it certainly isn't a single blob. This suggests that further investigation would be fruitful.*

the nearest kilometer. I then used principal coordinate analysis to find positions for each capital, and rotated, translated and scaled the resulting plot to check it against a real map (Figure 11.30).

One natural use of principal coordinate analysis is to see if one can spot any structure in a dataset. Does the dataset form a blob, or is it clumpy? This isn't a perfect test, but it's a good way to look and see if anything interesting is happening. In figure 11.31, I show a 3D plot of the spectral data, reduced to three dimensions using principal coordinate analysis. The plot is quite interesting. You should notice that the data points are spread out in 3D, but actually seem to lie on a complicated curved surface — they very clearly don't form a uniform blob. To me, the structure looks somewhat like a butterfly. I don't know why this occurs, but it certainly suggests that something worth investigating is going on. Perhaps the choice of samples that were measured is funny; perhaps the measuring instrument doesn't make certain kinds of measurement; or perhaps there are physical processes that prevent the data from spreading out over the space.

Our algorithm has one really interesting property. In some cases, we do not actually know the datapoints as vectors. Instead, we *just* know distances between the datapoints. This happens often in the social sciences, but there are important cases in computer science as well. As a rather contrived example, one could survey people about breakfast foods (say, eggs, bacon, cereal, oatmeal, pancakes, toast, muffins, kippers and sausages for a total of 9 items). We ask each person to rate the similarity of each pair of distinct items on some scale. We advise people that similar items are ones where, if they were offered both, they would have no particular preference; but, for dissimilar items, they would have a strong preference for one over the other. The scale might be "very similar", "quite similar", "similar", "quite dissimilar", and "very dissimilar" (scales like this are often called **Likert scales**). We collect these similarities from many people for each pair of distinct items, and then average the similarity over all respondents. We compute distances from the similarities in a way that makes very similar items close and very dissimilar items

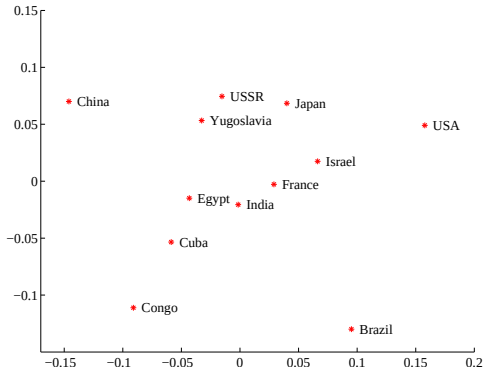

FIGURE 11.32: *A map of country similarity, prepared from the data of figure* **??**. *The map is often interpreted as showing a variation in development or wealth (poorest at bottom left to richest at top right); and freedom (most repressed at top left and freeest at bottom right). I haven't plotted these axes, because the interpretation wouldn't be consistent with current intuition (the similarity data is forty years old, and quite a lot has happened in that time).*

distant. Now we have a table of distances between items, and can compute a $\mathbf{V}$ and produce a scatter plot. This plot is quite revealing, because items that most people think are easily substituted appear close together, and items that are hard to substitute are far apart. The neat trick here is that we did not start with a $\mathcal{X}$, but with just a set of distances; but we were able to associate a vector with "eggs", and produce a meaningful plot.

Table **??** shows data from one such example. Students were interviewed (in 1971! things may have changed since then) about their perceptions of the similarity of countries. The averaged perceived similarity is shown in table **??**. Large numbers reflect high similarity, so we can't use these numbers directly. It is reasonable to turn these numbers into distances by (a) using 0 as the distance between a country and itself and (b) using $e^{-s_{ij}}$ as the distance between countries i and j (where s_{ij} is the similarity between them). Once we have distances, we can apply the procedure of section 11.4.1 to get points, then plot a scatter plot (Figure 11.32).

11.5 EXAMPLE: UNDERSTANDING HEIGHT AND WEIGHT

Recall the height-weight data set of section **??** (from `http://www2.stetson.edu/~jrasp/data.htm`; look for bodyfat.xls at that URL). This is, in fact, a 16-dimensional dataset. The entries are (in this order): *bodyfat; density; age; weight; height; adiposity; neck; chest; abdomen; hip; thigh; knee; ankle; biceps; forearm; wrist.* We know already that many of these entries are correlated, but it's hard to grasp a 16 dimensional dataset in one go. The first step is to investigate with a multidimensional scaling.

Figure **??** shows a multidimensional scaling of this dataset down to three dimensions. The dataset seems to lie on a (fairly) flat structure in 3D, meaning that inter-point distances are relatively well explained by a 2D representation. Two

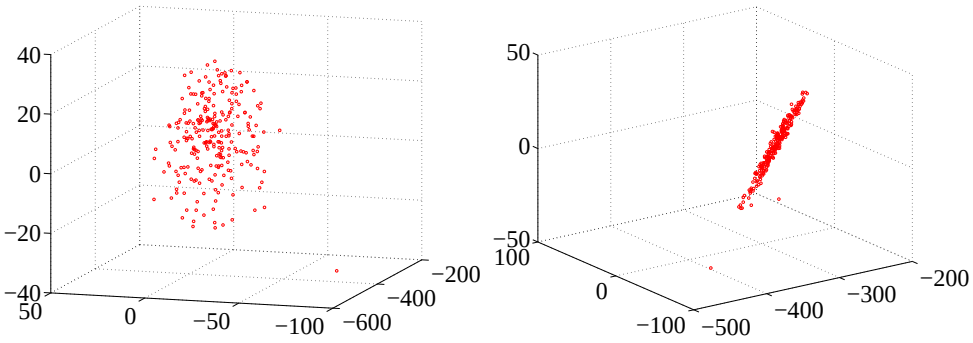

FIGURE 11.33: *Two views of a multidimensional scaling to three dimensions of the height-weight dataset. Notice how the data seems to lie in a flat structure in 3D, with one outlying data point. This means that the distances between data points can be (largely) explained by a 2D representation.*

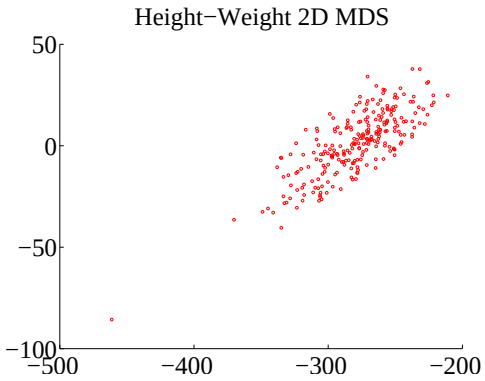

FIGURE 11.34: *A multidimensional scaling to two dimensions of the height-weight dataset. One data point is clearly special, and another looks pretty special. The data seems to form a blob, with one axis quite a lot more important than another.*

points seem to be special, and lie far away from the flat structure. The structure isn't perfectly flat, so there will be small errors in a 2D representation; but it's clear that a lot of dimensions are redundant. Figure 11.34 shows a 2D representation of these points. They form a blob that is stretched along one axis, and there is no sign of multiple blobs. There's still at least one special point, which we shall ignore but might be worth investigating further. The distortions involved in squashing this dataset down to 2D seem to have made the second special point less obvious than it was in figure **??**.

The next step is to try a principal component analysis. Figure 11.35 shows the mean of the dataset. The components of the dataset have different units, and shouldn't really be compared. But it is difficult to interpret a table of 16 numbers, so I have plotted the mean by showing a vertical bar for each component. Figure 11.36 shows the eigenvalues of the covariance for this dataset. Notice how one dimension is

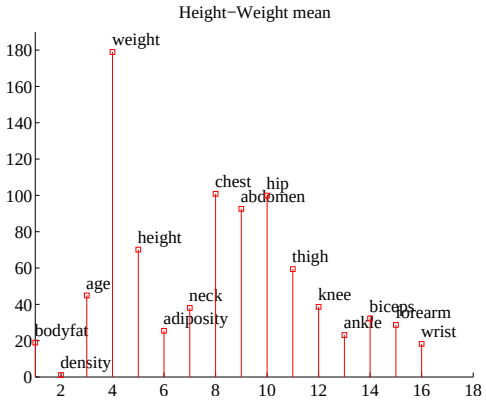

FIGURE 11.35: *The mean of the bodyfat.xls dataset. Each component is likely in a different unit (though I don't know the units), making it difficult to plot the data without being misleading. I've adopted one solution here, by plotting each component as a vertical bar, and labelling the bar. You shouldn't try to compare the values to one another. Instead, think of this plot as a compact version of a table.*

very important, and after the third principal component, the contributions become small. Of course, I could have said "fourth", or "fifth", or whatever — the precise choice depends on how small a number you think is "small".

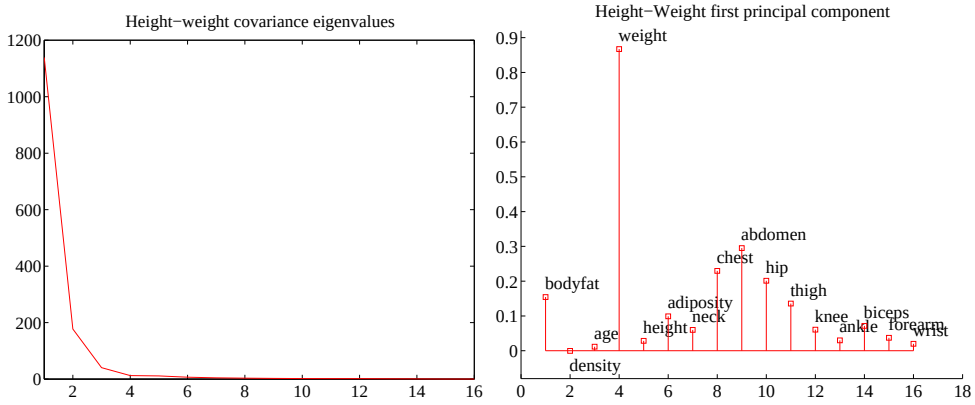

FIGURE 11.36: *On the* **left**, *the eigenvalues of the covariance matrix for the bodyfat data set. Notice how fast the eigenvalues fall off; this means that most principal components have very small variance, so that data can be represented well with a small number of principal components. On the* **right**, *the first principal component for this dataset, plotted using the same convention as for figure 11.35.*

Figure 11.36 also shows the first principal component. The eigenvalues justify thinking of each data item as (roughly) the mean plus some weight times this principal component. From this plot you can see that data items with a larger

value of *weight* will also have larger values of most other measurements, except *age* and *density*. You can also see how much larger; if the weight goes up by 8.5 units, then the abdomen will go up by 3 units, and so on. This explains the main variation in the dataset.

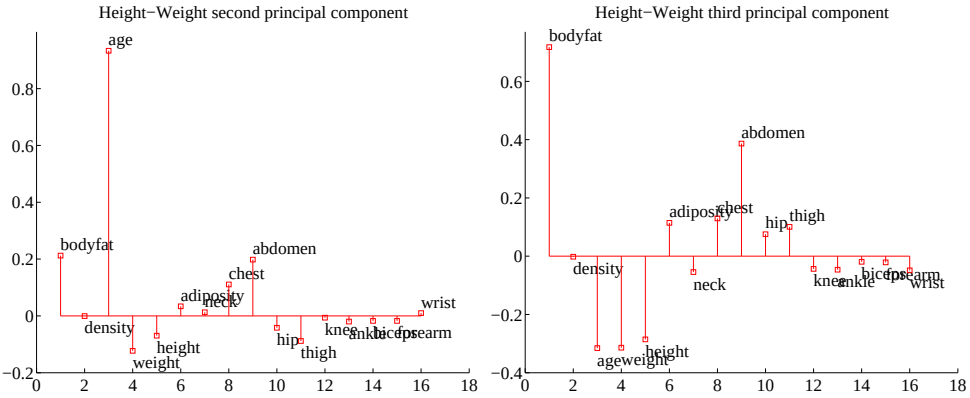

FIGURE 11.37: *On the* **left**, *the second principal component, and on the* **right** *the third principal component of the height-weight dataset.*

In the rotated coordinate system, the components are not correlated, and they have different variances (which are the eigenvalues of the covariance matrix). You can get some sense of the data by adding these variances; in this case, we get 1404. This means that, in the translated and rotated coordinate system, the average data point is about $37 = \sqrt{1404}$ units away from the center (the origin). Translations and rotations do not change distances, so the average data point is about 37 units from the center in the original dataset, too. If we represent a datapoint by using the mean and the first three principal components, there will be some error. We can estimate the average error from the component variances. In this case, the sum of the first three eigenvalues is 1357, so the mean square error in representing a datapoint by the first three principal components is $\sqrt{(1404 - 1357)}$, or 6.8. The relative error is $6.8/37 = 0.18$. Another way to represent this information, which is more widely used, is to say that the first three principal components explain all but $(1404 - 1357)/1404 = 0.034$, or 3.4% of the variance; notice that this is the square of the relative error, which will be a much smaller number.

All this means that explaining a data point as the mean and the first three principal components produces relatively small errors. Figure 11.38 shows the second and third principal component of the data. These two principal components suggest some further conclusions. As *age* gets larger, *height* and *weight* get slightly smaller, but the weight is redistributed; *abdomen* gets larger, whereas *thigh* gets smaller. A smaller effect (the third principal component) links *bodyfat* and *abdomen*. As *bodyfat* goes up, so does *abdomen*.

11.6 WHAT YOU SHOULD REMEMBER - NEED SOMETHING

PROBLEMS

Summaries

11.1. You have a dataset $\{\mathbf{x}\}$ of N vectors, $\mathbf{x}_i$, each of which is d-dimensional. We will consider a linear function of this dataset. Write $\mathbf{a}$ for a constant vector; then the value of this linear function evaluated on the i'th data item is $\mathbf{a}^T\mathbf{x}_i$. Write $f_i = \mathbf{a}^T\mathbf{x}_i$. We can make a new dataset $\{f\}$ out of the values of this linear function.

(a) Show that $\mathsf{mean}(\{f\}) = \mathbf{a}^T\,\mathsf{mean}(\{\mathbf{x}\})$ (easy).

(b) Show that $\mathsf{var}(\{f\}) = \mathbf{a}^T\,\mathsf{Covmat}(\{\mathbf{x}\})\mathbf{a}$ (harder, but just push it through the definition).

(c) Assume the dataset has the special property that there exists some $\mathbf{a}$ so that $\mathbf{a}^T\,\mathsf{Covmat}(\{\mathbf{x}\})\mathbf{a}$. Show that this means that the dataset lies on a hyperplane.

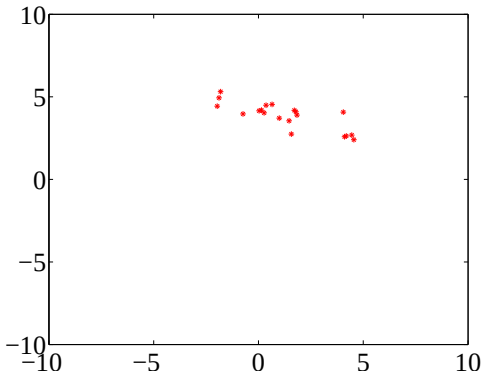

FIGURE 11.38: *Figure for the question*

11.2. On Figure 11.38, mark the mean of the dataset, the first principal component, and the second principal component.

11.3. You have a dataset $\{\mathbf{x}\}$ of N vectors, $\mathbf{x}_i$, each of which is d-dimensional. Assume that $\mathsf{Covmat}(\{\mathbf{x}\})$ has one non-zero eigenvalue. Assume that $\mathbf{x}_1$ and $\mathbf{x}_2$ do not have the same value.

(a) Show that you can choose a set of t_i so that you can represent *every* data item $\mathbf{x}_i$ *exactly*

$$\mathbf{x}_i = \mathbf{x}_1 + t_i(\mathbf{x}_2 - \mathbf{x}_1).$$

(b) Now consider the dataset of these t values. What is the relationship between (a) $\mathsf{std}(t)$ and (b) the non-zero eigenvalue of $\mathsf{Covmat}(\{\mathbf{x}\})$? Why?

CHAPTER 12

Clustering: Models of High Dimensional Data

High-dimensional data comes with problems. Data points tend not to be where you think; they can scattered quite far apart, and can be quite far from the mean. Except in special cases, the only really reliable probability model is the Gaussian (or Gaussian blob, or blob).

There is an important rule of thumb for coping with high dimensional data: **Use simple models**. A blob is another good, simple model. Modelling data as a blob involves computing its mean and its covariance. Sometimes, as we shall see, the covariance can be hard to compute. Even so, a blob model is really useful. It is natural to try and extend this model to cover datasets that don't obviously consist of a single blob.

One very good, very simple, model for high dimensional data is to assume that it consists of multiple blobs. To build models like this, we must determine (a) what the blob parameters are and (b) which datapoints belong to which blob. Generally, we will collect together data points that are close and form blobs out of them. This process is known as **clustering**.

Clustering is a somewhat puzzling activity. It is extremely useful to cluster data, and it seems to be quite important to do it reasonably well. But it surprisingly hard to give crisp criteria for a good (resp. bad) clustering of a dataset. Usually, clustering is part of building a model, and the main way to know that the clustering algorithm is bad is that the model is bad.

12.1 THE CURSE OF DIMENSION

High dimensional models display uninituitive behavior (or, rather, it can take years to make your intuition see the true behavior of high-dimensional models as natural). In these models, most data lies in places you don't expect. We will do several simple calculations with an easy high-dimensional distribution to build some intuition.

Assume our data lies within a cube, with edge length two, centered on the origin. This means that each component of $\mathbf{x}_i$ lies in the range $[-1, 1]$. One simple model for such data is to assume that each dimension has uniform probability density in this range. In turn, this means that $P(x) = \frac{1}{2^d}$. The mean of this model is at the origin, which we write as $\mathbf{0}$.

The first surprising fact about high dimensional data is that most of the data can lie quite far away from the mean. For example, we can divide our dataset into two pieces. $\mathcal{A}(\epsilon)$ consists of all data items where *every* component of the data has a value in the range $[-(1-\epsilon), (1-\epsilon)]$. $\mathcal{B}(\epsilon)$ consists of all the rest of the data. If you think of the data set as forming a cubical orange, then $\mathcal{B}(\epsilon)$ is the rind (which has thickness ϵ) and $\mathcal{A}(\epsilon)$ is the fruit.

Your intuition will tell you that there is more fruit than rind. This is true,

for three dimensional oranges, but not true in high dimensions. The fact that the orange is cubical just simplifies the calculations, but has nothing to do with the real problem.

We can compute $P(\{\mathbf{x} \in \mathcal{A}(\epsilon)\})$ and $P(\{\mathbf{x} \in \mathcal{A}(\epsilon)\})$. These probabilities tell us the probability a data item lies in the fruit (resp. rind). $P(\{\mathbf{x} \in \mathcal{A}(\epsilon)\})$ is easy to compute as

$$P(\{\mathbf{x} \in \mathcal{A}(\epsilon)\}) = (2(1-\epsilon)))^d \left(\frac{1}{2^d}\right) = (1-\epsilon)^d$$

and

$$P(\{\mathbf{x} \in \mathcal{B}(\epsilon)\}) = 1 - P(\{\mathbf{x} \in \mathcal{A}(\epsilon)\}) = 1 - (1-\epsilon)^d.$$

But notice that, as $d \to \infty$,

$$P(\{\mathbf{x} \in \mathcal{A}(\epsilon)\}) \to 0.$$

This means that, for large d, we expect most of the data to be in $\mathcal{B}(\epsilon)$. Equivalently, for large d, we expect that at least one component of each data item is close to either 1 or -1.

This suggests (correctly) that much data is quite far from the origin. It is easy to compute the average of the squared distance of data from the origin. We want

$$\mathbb{E}\left[\mathbf{x}^T\mathbf{x}\right] = \int_{\text{box}} \left(\sum_i x_i^2\right) P(\mathbf{x}) d\mathbf{x}$$

but we can rearrange, so that

$$\mathbb{E}\left[\mathbf{x}^T\mathbf{x}\right] = \sum_i \mathbb{E}\left[x_i^2\right] = \sum_i \int_{\text{box}} x_i^2 P(\mathbf{x}) d\mathbf{x}.$$

Now each component of $\mathbf{x}$ is independent, so that $P(\mathbf{x}) = P(x_1)P(x_2)\dots P(x_d)$. Now we substitute, to get

$$\mathbb{E}\left[\mathbf{x}^T\mathbf{x}\right] = \sum_i \mathbb{E}\left[x_i^2\right] = \sum_i \int_{-1}^{1} x_i^2 P(x_i) dx_i = \sum_i \frac{1}{2} \int_{-1}^{1} x_i^2 dx_i = \frac{d}{3},$$

so as d gets bigger, most data points will be further and further from the origin. Worse, as d gets bigger, data points tend to get further and further from one another. We can see this by computing the average of the squared distance of data points from one another. Write $\mathbf{u}$ for one data point and $\mathbf{v}$; we can compute

$$\mathbb{E}\left[d(\mathbf{u},\mathbf{v})^2\right] = \int_{\text{box}} \int_{\text{box}} \sum_i (u_i - v_i)^2 d\mathbf{u} d\mathbf{v} = \mathbb{E}\left[\mathbf{u}^T\mathbf{u}\right] + \mathbb{E}\left[\mathbf{v}^T\mathbf{v}\right] - \mathbb{E}\left[\mathbf{u}^T\mathbf{v}\right]$$

but since $\mathbf{u}$ and $\mathbf{v}$ are independent, we have $\mathbb{E}\left[\mathbf{u}^T\mathbf{v}\right] = \mathbb{E}[\mathbf{u}]^T\mathbb{E}[\mathbf{v}] = 0$. This yields

$$\mathbb{E}\left[d(\mathbf{u},\mathbf{v})^2\right] = 2\frac{d}{3}$$

meaning that, for large d, we expect our data points to be quite far apart.

It is difficult to build histogram representations for high dimensional data. The strategy of dividing the domain into boxes, then counting data into them, fails miserably because there are too many boxes. In the case of our cube, imagine we wish to divide each dimension in half (i.e. between $[-1, 0]$ and between $[0, 1]$). Then we must have 2^d boxes. This presents two problems. First, we will have difficulty representing this number of boxes. Second, unless we are exceptionally lucky, most boxes must be empty because we will not have 2^d data items.

However, one representation is extremely effective. We can represent data as a collection of **clusters** — coherent blobs of similar datapoints that could, under appropriate circumstances, be regarded as the same. We discuss how to build these clusters, then what to do with them.

12.2 AGGLOMERATIVE AND DIVISIVE CLUSTERING

There are two natural algorithms for clustering. In **divisive clustering**, the entire data set is regarded as a cluster, and then clusters are recursively split to yield a good clustering (Algorithm 12.2). In **agglomerative clustering**, each data item is regarded as a cluster, and clusters are recursively merged to yield a good clustering (Algorithm 12.1).

```
Make each point a separate cluster
Until the clustering is satisfactory
    Merge the two clusters with the
      smallest inter-cluster distance
end
```

Algorithm 12.1: *Agglomerative Clustering or Clustering by Merging.*

```
Construct a single cluster containing all points
Until the clustering is satisfactory
    Split the cluster that yields the two
      components with the largest inter-cluster distance
end
```

Algorithm 12.2: *Divisive Clustering, or Clustering by Splitting.*

There are two major issues in thinking about clustering:

- *What is a good inter-cluster distance?* Agglomerative clustering uses an inter-cluster distance to fuse nearby clusters; divisive clustering uses it to split insufficiently coherent clusters. Even if a natural distance between data points is available (which might not be the case for vision problems), there is no canonical inter-cluster distance. Generally, one chooses a distance that seems appropriate for the data set. For example, one might choose the distance

between the closest elements as the inter-cluster distance, which tends to yield extended clusters (statisticians call this method **single-link clustering**). Another natural choice is the maximum distance between an element of the first cluster and one of the second, which tends to yield rounded clusters (statisticians call this method **complete-link clustering**). Finally, one could use an average of distances between elements in the cluster, which also tends to yield "rounded" clusters (statisticians call this method **group average clustering**).

- *How many clusters are there?* This is an intrinsically difficult task if there is no model for the process that generated the clusters. The algorithms we have described generate a hierarchy of clusters. Usually, this hierarchy is displayed to a user in the form of a **dendrogram**—a representation of the structure of the hierarchy of clusters that displays inter-cluster distances—and an appropriate choice of clusters is made from the dendrogram (see the example in Figure 12.1).

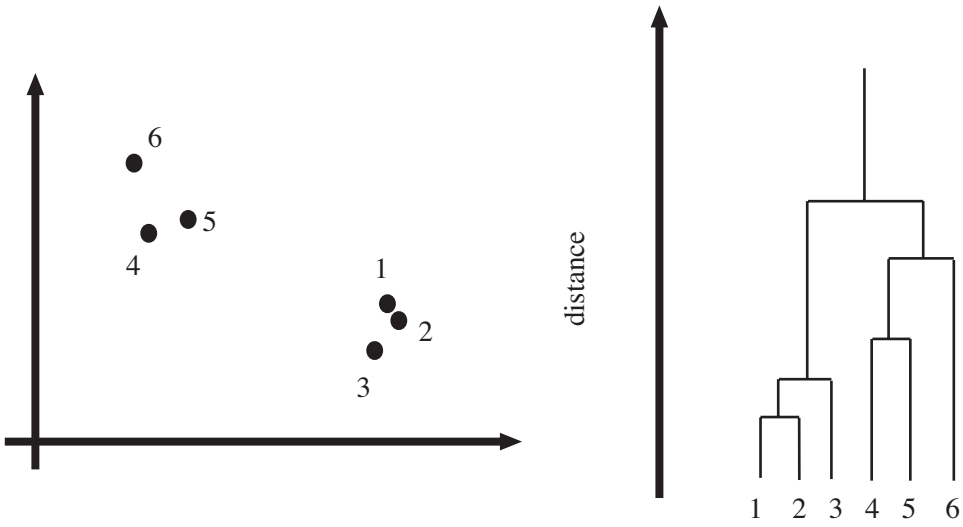

FIGURE 12.1: **Left**, *a data set;* **right**, *a dendrogram obtained by agglomerative clustering using single-link clustering. If one selects a particular value of distance, then a horizontal line at that distance splits the dendrogram into clusters. This representation makes it possible to guess how many clusters there are and to get some insight into how good the clusters are.*

The main difficulty in using a divisive model is knowing where to split. This is sometimes made easier for particular kinds of data. For example, we could segment an image by clustering pixel values. In this case, you can sometimes find good splits by constructing a histogram of intensities, or of color values.

Another important thing to notice about clustering from the example of figure 12.1 is that there is no right answer. There are a variety of different clusterings

of the same data. For example, depending on what scales in that figure mean, it might be right to zoom out and regard all of the data as a single cluster, or to zoom in and regard each data point as a cluster. Each of these representations may be useful.

12.2.1 Clustering and Distance

In the algorithms above, and in what follows, we assume that the features are scaled so that distances (measured in the usual way) between data points are a good representation of their similarity. This is quite an important point. For example, imagine we are clustering data representing brick walls. The features might contain several distances: the spacing between the bricks, the length of the wall, the height of the wall, and so on. If these distances are given in the same set of units, we could have real trouble. For example, assume that the units are centimeters. Then the spacing between bricks is of the order of one or two centimeters, but the heights of the walls will be in the hundreds of centimeters. In turn, this means that the distance between two datapoints is likely to be completely dominated by the height and length data. This could be what we want, but it might also not be a good thing.

There are some ways to manage this issue. One is to know what the features measure, and know how they should be scaled. Usually, this happens because you have a deep understanding of your data. If you don't (which happens!), then it is often a good idea to try and normalize the scale of the data set. There are two good strategies. The simplest is to translate the data so that it has zero mean (this is just for neatness - translation doesn't change distances), then scale each direction so that it has unit variance. More sophisticated is to translate the data so that it has zero mean, then transform it so that each direction is independent and has unit variance. Doing so is sometimes referred to as **decorrelation** or **whitening** (because you make the data more like white noise).

TODO: no - fix the whitening link here

TODO: show how in R?

TODO: worked example in R?

12.2.2 Example: Agglomerative Clustering

Matlab provides some tools that are useful for agglomerative clustering. These functions use a scheme where one first builds the whole tree of merges, then analyzes that tree to decide which clustering to report. `linkage` will determine which pairs of clusters should be merged at which step (there are arguments that allow you to choose what type of inter-cluster distance it should use); `dendrogram` will plot you a dendrogram; and `cluster` will extract the clusters from the linkage, using a variety of options for choosing the clusters. I used these functions to prepare the dendrogram of figure 12.2 for the height-weight dataset of section **??** (from `http://www2.stetson.edu/~jrasp/data.htm`; look for bodyfat.xls). I deliberately forced Matlab to plot the whole dendrogram, which accounts for the crowded look of the

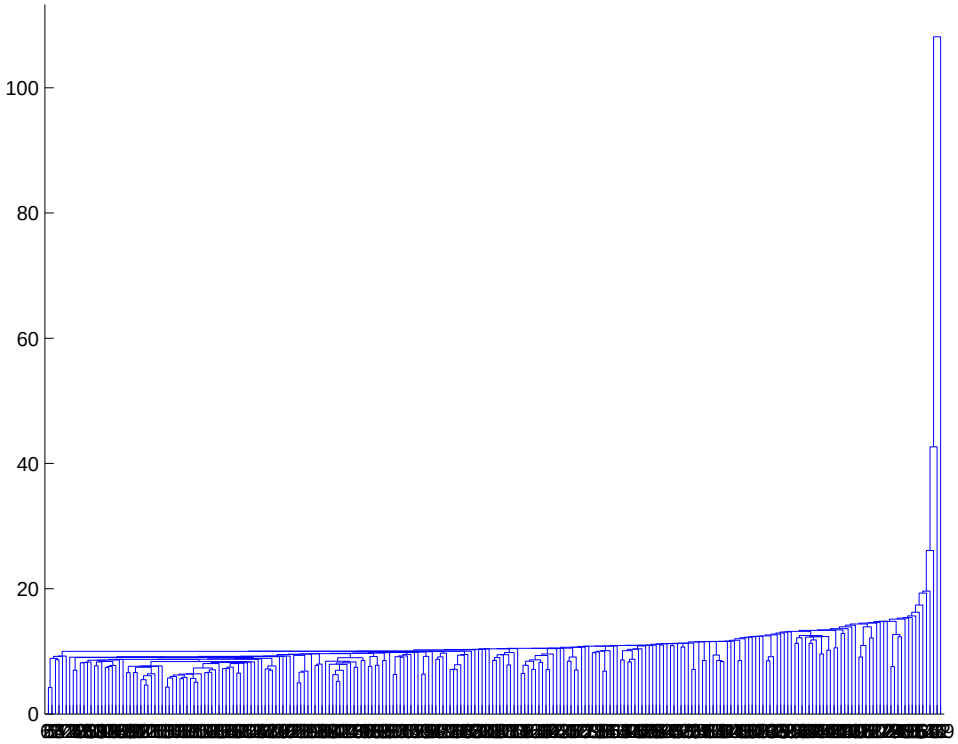

FIGURE 12.2: *A dendrogram obtained from the body fat dataset, using single link clustering. Recall that the data points are on the horizontal axis, and that the vertical axis is distance; there is a horizontal line linking two clusters that get merged, established at the height at which they're merged. I have plotted the entire dendrogram, despite the fact it's a bit crowded at the bottom, because it shows that most data points are relatively close (i.e. there are lots of horizontal branches at about the same height).*

figure (you can allow it to merge small leaves, etc.). I used a single-link strategy. In particular, notice that many data points are about the same distance from one another, which suggests a single big cluster with a smaller set of nearby clusters. The clustering of figure 12.3 supports this view. I plotted the data points on the first two principal components, using different colors and shapes of marker to indicate different clusters. There are a total of 30 clusters here, though most are small.

As another example, I obtained the seed dataset from the UC Irvine Machine Learning Dataset Repository (you can find it at `http://archive.ics.uci.edu/ml/datasets/seeds`). Each item consists of seven measurements of a wheat kernel; there are three types of wheat represented in this dataset. As you can see in figures 12.4 and 12.5, this data clusters rather well.

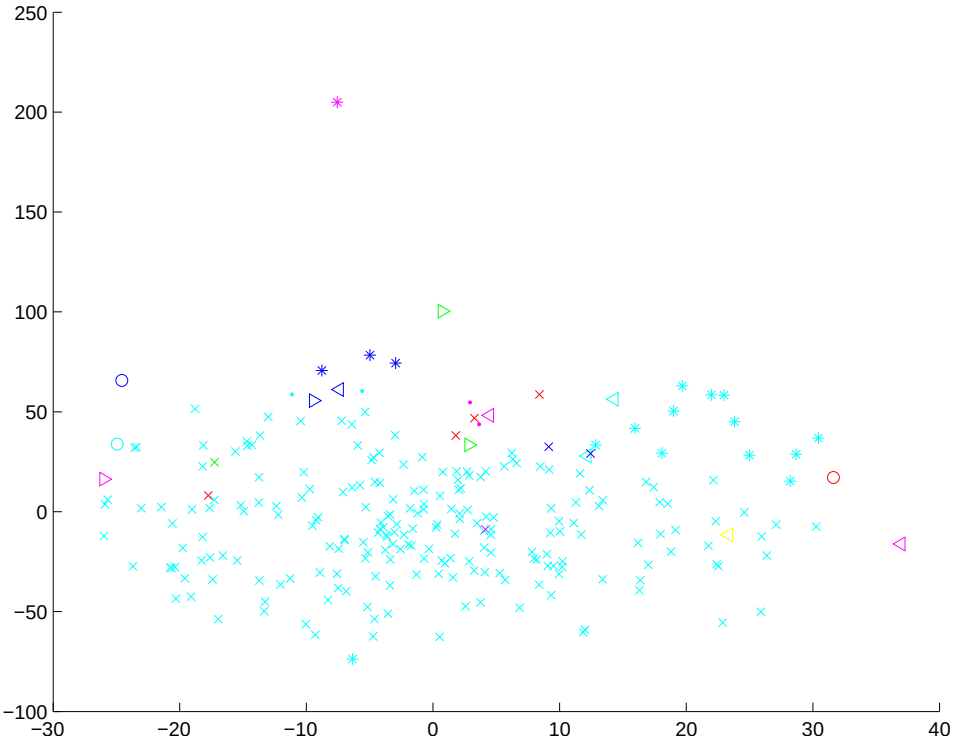

FIGURE 12.3: *A clustering of the body fat dataset, using agglomerative clustering, single link distance, and requiring a maximum of 30 clusters. I have plotted each cluster with a distinct marker (though some markers differ only by color; you might need to look at the PDF version to see this figure at its best). Notice that one cluster contains much of the data, and that there are a set of small isolated clusters. The original data is 16 dimensional, which presents plotting problems; I show a scatter plot on the first two principal components (though I computed distances for clustering in the original 16 dimensional space).*

12.3 THE K-MEANS ALGORITHM AND VARIANTS

TODO: Figure notes: Iris figure, estimate of k, 2d iris plot, projected onto pca plot, another cluster dataset

Assume we have a dataset that, we believe, forms many clusters that look like blobs. If we knew where the center of each of the clusters was, it would be easy to tell which cluster each data item belonged to — it would belong to the cluster with the closest center. Similarly, if we knew which cluster each data item belonged to, it would be easy to tell where the cluster centers were — they'd be the mean of the data items in the cluster. This is the point closest to every point in the cluster.

We can turn these observations into an algorithm. Assume that we know how many clusters there are in the data, and write k for this number. The jth data

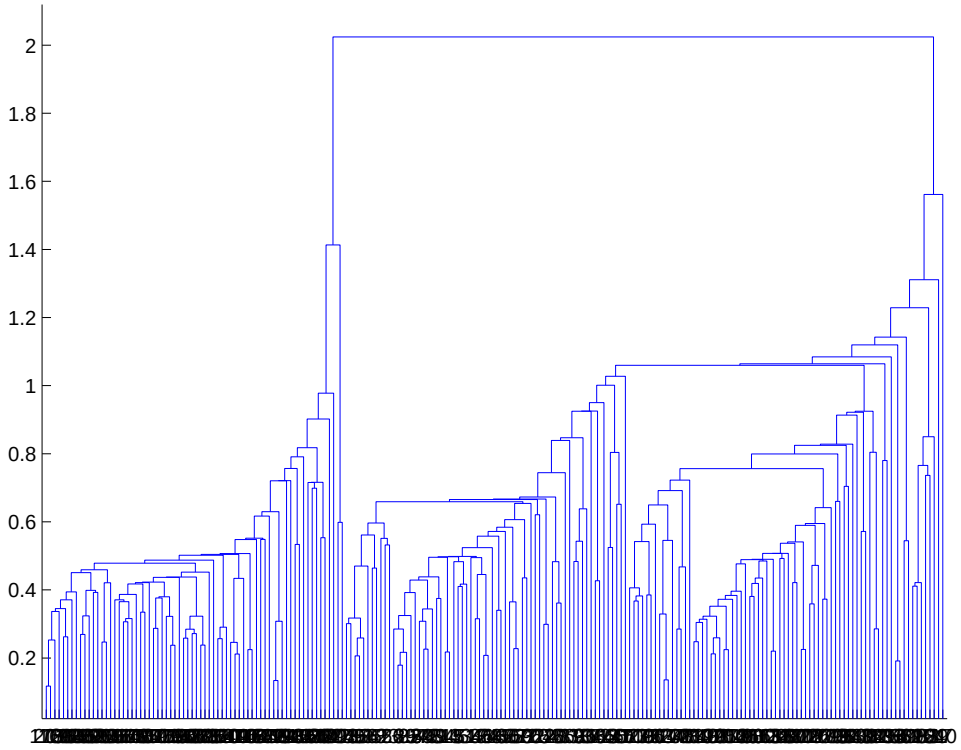

FIGURE 12.4: *A dendrogram obtained from the seed dataset, using single link clustering. Recall that the data points are on the horizontal axis, and that the vertical axis is distance; there is a horizontal line linking two clusters that get merged, established at the height at which they're merged. I have plotted the entire dendrogram, despite the fact it's a bit crowded at the bottom, because you can now see how clearly the data set clusters into a small set of clusters — there are a small number of vertical "runs".*

item to be clustered is described by a feature vector $\mathbf{x}_j$. We write $\mathbf{c}_i$ for the center of the ith cluster. We assume that most data items are close to the center of their cluster. This suggests that we cluster the data by minimizing the the cost function

$$\Phi(\text{clusters}, \text{data}) = \sum_{i \in \text{clusters}} \left\{ \sum_{j \in i\text{th cluster}} (\mathbf{x}_j - \mathbf{c}_i)^T (\mathbf{x}_j - \mathbf{c}_i) \right\}.$$

Notice that if we know the center for each cluster, it is easy to determine which cluster is the best choice for each point. Similarly, if the allocation of points to clusters is known, it is easy to compute the best center for each cluster. However, there are far too many possible allocations of points to clusters to search this space for a minimum. Instead, we define an algorithm that iterates through two activities:

- Assume the cluster centers are known and, allocate each point to the closest

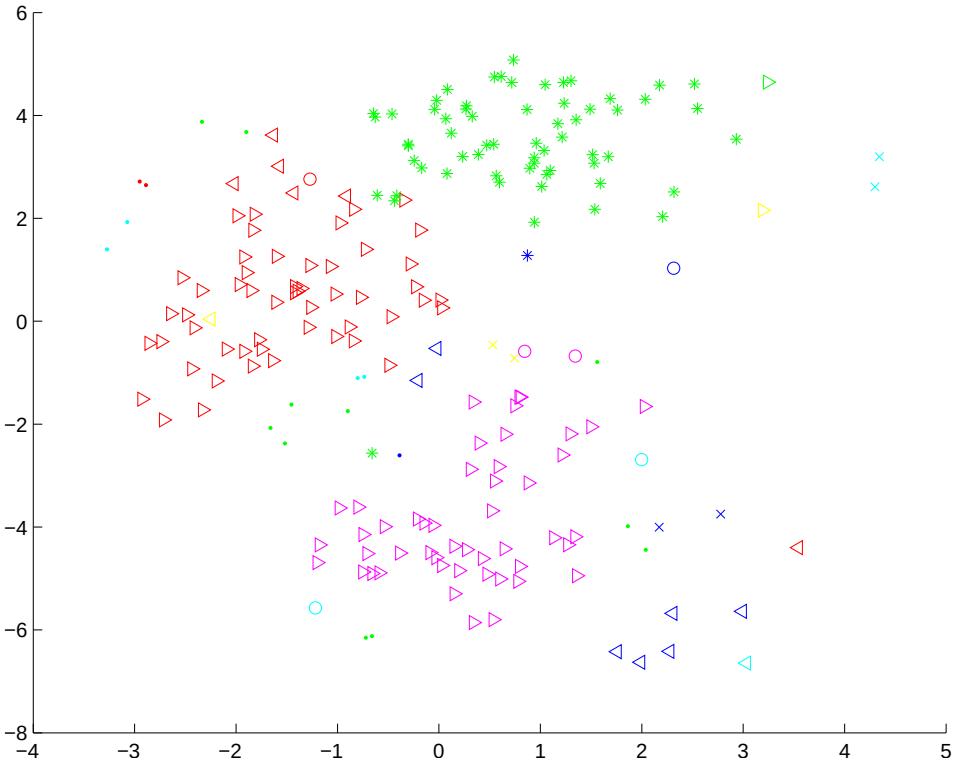

FIGURE 12.5: *A clustering of the seed dataset, using agglomerative clustering, single link distance, and requiring a maximum of 30 clusters. I have plotted each cluster with a distinct marker (though some markers differ only by color; you might need to look at the PDF version to see this figure at its best). Notice that there are a set of fairly natural isolated clusters. The original data is 8 dimensional, which presents plotting problems; I show a scatter plot on the first two principal components (though I computed distances for clustering in the original 8 dimensional space).*

cluster center.

- Assume the allocation is known, and choose a new set of cluster centers. Each center is the mean of the points allocated to that cluster.

We then choose a start point by randomly choosing cluster centers, and then iterate these stages alternately. This process eventually converges to a local minimum of the objective function (the value either goes down or is fixed at each step, and it is bounded below). It is not guaranteed to converge to the global minimum of the objective function, however. It is also not guaranteed to produce k clusters, unless we modify the allocation phase to ensure that each cluster has some nonzero number of points. This algorithm is usually referred to as **k-means** (summarized in Algorithm 12.3).

Usually, we are clustering high dimensional data, so that visualizing clusters

```
Choose k data points to act as cluster centers
Until the cluster centers change very little
    Allocate each data point to cluster whose center is nearest.
    Now ensure that every cluster has at least
      one data point; one way to do this is by
      supplying empty clusters with a point chosen at random from
      points far from their cluster center.
    Replace the cluster centers with the mean of the elements
      in their clusters.
end
```

Algorithm 12.3: *Clustering by K-Means.*

can present a challenge. If the dimension isn't too high, then we can use panel plots. An alternative is to project the data onto two principal components, and plot the clusters there. A natural dataset to use to explore k-means is the iris data, where we know that the data should form three clusters (because there are three species). Recall this dataset from section **??**. I reproduce figure 11.6 from that section, for comparison. Figures 12.6, **??** and **??** show different k-means clusterings of that data.

12.3.1 How to choose K

The iris data is just a simple example. We know that the data forms clean clusters, and we know there should be three of them. Usually, we don't know how many clusters there should be, and we need to choose this by experiment. One strategy is to cluster for a variety of different values of k, then look at the value of the cost function for each. If there are more centers, each data point can find a center that is close to it, so we expect the value to go down as k goes up. This means that looking for the k that gives the smallest value of the cost function is not helpful, because that k is always the same as the number of data points (and the value is then zero). However, it can be very helpful to plot the value as a function of k, then look at the "knee" of the curve. Figure 12.9 shows this plot for the iris data. Notice that $k = 3$ — the "true" answer — doesn't look particularly special, but $k = 2$, $k = 3$, or $k = 4$ all seem like reasonable choices. It is possible to come up with a procedure that makes a more precise recommendation by penalizing clusterings that use a large k, because they may represent inefficient encodings of the data. However, this is often not worth the bother.

In some special cases (like the iris example), we might know the right answer to check our clustering against. In such cases, one can evaluate the clustering by looking at the number of different labels in a cluster (sometimes called the purity), and the number of clusters. A good solution will have few clusters, all of which have high purity.

Mostly, we don't have a right answer to check against. An alternative strategy, which might seem crude to you, for choosing k is extremely important in practice.

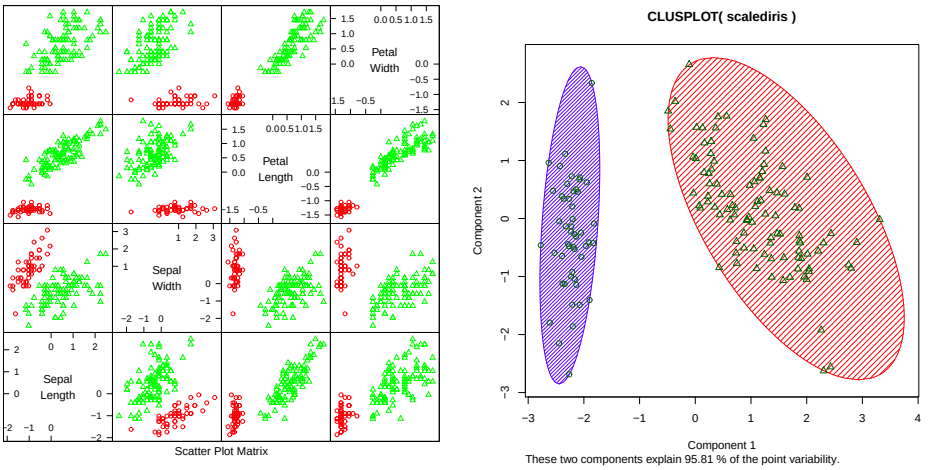

FIGURE 12.6: *On the* **left**, *a panel plot of the iris data clustered using k-means with* $k = 2$. *By comparison with figure 12.9, notice how the* versicolor *and* verginica *clusters appear to have been merged. On the* **right**, *this data set projected onto the first two principal components, with one blob drawn over each cluster.*

Usually, one clusters data to use the clusters in an application (one of the most important, vector quantization, is described in section 16). There are usually natural ways to evaluate this application. For example, vector quantization is often used as an early step in texture recognition or in image matching; here one can evaluate the error rate of the recognizer, or the accuracy of the image matcher. One then chooses the k that gets the best evaluation score on validation data. In this view, the issue is not how good the clustering is; it's how well the system that uses the clustering works.

12.3.2 Soft Assignment

One difficulty with k-means is that each point must belong to exactly one cluster. But, given we don't know how many clusters there are, this seems wrong. If a point is close to more than one cluster, why should it be forced to choose? This reasoning suggests we assign points to cluster centers with weights.

We allow each point to carry a total weight of 1. In the conventional k-means algorithm, it must choose a single cluster, and assign its weight to that cluster alone. In soft-assignment k-means, the point can allocate some weight to each cluster center, as long as (a) the weights are all non-negative and (b) the weights sum to one. Write $w_{i,j}$ for the weight connecting point i to cluster center j. We interpret these weights as the degree to which the point participates in a particular cluster. We require $w_{i,j} \geq 0$ and $\sum_j w_{i,j} = 1$.

We would like $w_{i,j}$ to be large when $\mathbf{x}_i$ is close to $\mathbf{c}_j$, and small otherwise. Write $d_{i,j}$ for the distance $\| \mathbf{x}_i - \mathbf{c}_j \|$ between these two. Write

$$s_{i,j} = e^{\frac{-d_{i,j}^2}{2\sigma^2}}$$

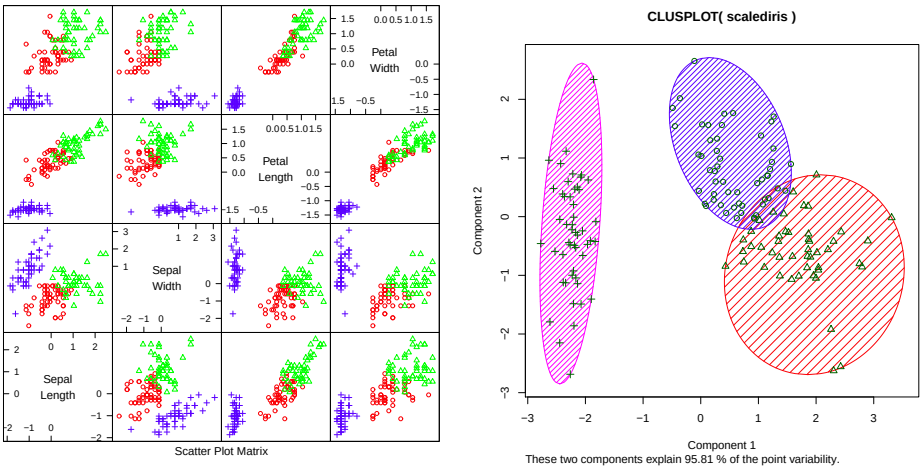

FIGURE 12.7: *On the* **left**, *a panel plot of the iris data clustered using k-means with* $k = 3$. *By comparison with figure 12.9, notice how the clusters appear to follow the species labels. On the* **right**, *this data set projected onto the first two principal components, with one blob drawn over each cluster.*

where $\sigma > 0$ is a choice of scaling parameter. This is often called the affinity between the point i and the center j. Now a natural choice of weights is

$$w_{i,j} = \frac{s_{i,j}}{\sum_{l=1}^{k} s_{i,l}}.$$

All these weights are non-negative, they sum to one, and the weight is large if the point is much closer to one center than to any other. The scaling parameter σ sets the meaning of "much closer" — we measure distance in units of σ.

Once we have weights, re-estimating the cluster centers is easy. We use a weights to compute a weighted average of the points. In particular, we re-estimate the j'th cluster center by

$$\frac{\sum_i w_{i,j}\mathbf{x}_i}{\sum_i w_{i,j}}.$$

Notice that k-means is a special case of this algorithm where σ limits to zero. In this case, each point has a weight of one for some cluster, and zero for all others, and the weighted mean becomes an ordinary mean. I have collected the description into Algorithm 12.4 for convenience.

12.3.3 General Comments on K-Means

If you experiment with k-means, you will notice one irritating habit of the algorithm. It almost always produces either some rather spread out clusters, or some single element clusters. Most clusters are usually rather tight and blobby clusters, but there is usually one or more bad cluster. This is fairly easily explained. Because every data point must belong to some cluster, data points that are far from all

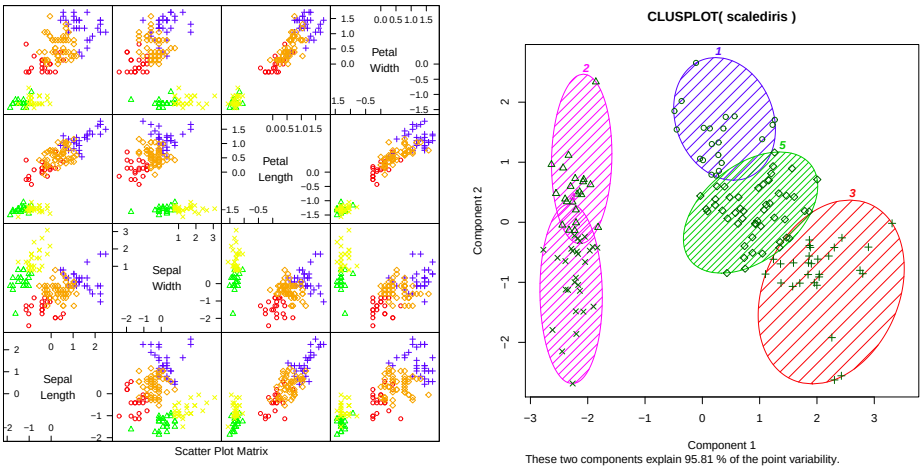

FIGURE 12.8: *On the* **left**, *a panel plot of the iris data clustered using k-means with* $k = 5$. *By comparison with figure 12.9, notice how* setosa *seems to have been broken in two groups, and* versicolor *and* verginica *into a total of three . On the* **right**, *this data set projected onto the first two principal components, with one blob drawn over each cluster.*

others (a) belong to some cluster and (b) very likely "drag" the cluster center into a poor location. This applies even if you use soft assignment, because every point must have total weight one. If the point is far from all others, then it will be assigned to the closest with a weight very close to one, and so may drag it into a poor location, or it will be in a cluster on its own.

There are ways to deal with this. If k is very big, the problem is often not significant, because then you simply have many single element clusters that you can ignore. It isn't always a good idea to have too large a k, because then some larger clusters might break up. An alternative is to have a junk cluster. Any point that is too far from the closest true cluster center is assigned to the junk cluster, and the center of the junk cluster is not estimated. Notice that points should not be assigned to the junk cluster permanently; they should be able to move in and out of the junk cluster as the cluster centers move.

In some cases, we want to cluster objects that can't be averaged. For example, you can compute distances between two trees but you can't meaningfully average them. In some cases, you might have a table of distances between objects, but not know vectors representing the objects. For example, one could collect data on the similarities between countries (as in Section 11.4.2, particularly Figure 11.32), then try and cluster using this data (similarities can be turned into distances by, for example, taking the negative logarithm). A variant of k-means, known as k-medoids, applies to this case.

In k-medoids, the cluster centers are data items rather than averages, but the rest of the algorithm has a familiar form. We assume the number of medoids is known, and initialize these randomly. We then iterate two procedures. In the first,

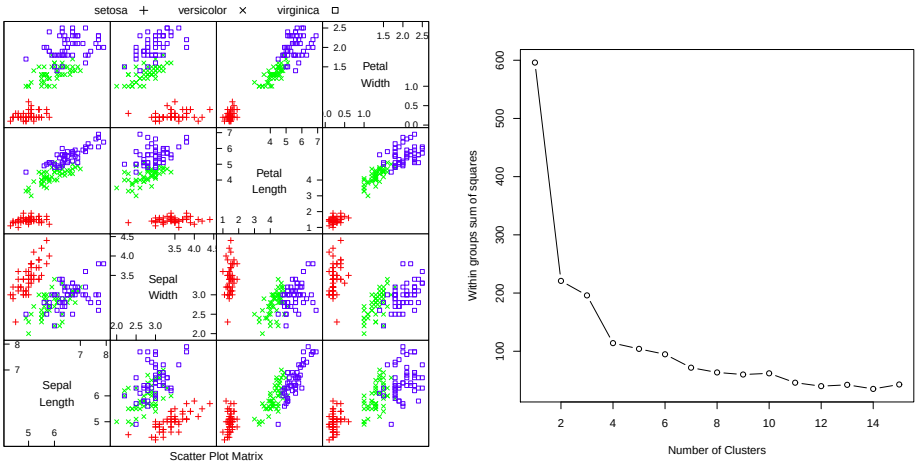

FIGURE 12.9: *On the* **left**, *the scatterplot matrix for the Iris data, for reference. On the* **right**, *a plot of the value of the cost function for each of several different values of k. Notice how there is a sharp drop in cost going from $k = 1$ to $k = 2$, and again at $k = 4$; after that, the cost falls off slowly. This suggests using $k = 2$, $k = 3$, or $k = 4$, depending on the precise application.*

we allocate data points to medoids. In the second, we choose the best medoid for each cluster by finding the medoid that minimizes the sum of distances of points in the cluster to that medoid (blank search is fine).

12.4 WHAT YOU SHOULD REMEMBER - NEED SOMETHING

Choose k data points to act as cluster centers
Until the cluster centers change very little
    First, we estimate the weights

    For i indexing data points
        For j indexing cluster centers
            Compute $s_{i,j} = e^{\frac{-\|\mathbf{x}_i - \mathbf{c}_j\|}{2\sigma^2}}$

    For i indexing data points
        For j indexing cluster centers
            Compute $w_{i,j} = s_{i,j} / \sum_{l=1} k s_{i,l}$

    Now, we re-estimate the centers

    For j indexing cluster centers
        Compute $\mathbf{c}_j = \frac{\sum_i w_{i,j} \mathbf{x}_i}{\sum_i w_{i,j}}$

end

Algorithm 12.4: *Soft Clustering by K-Means.*

CHAPTER 13

Regression

Classification tries to predict a class from a data item. Regression tries to predict a value. There are several reasons to do this. First, we might actually need a value. For example, we know the zip code of a house, the square footage of its lot, the number of rooms and the square footage of the house, and we wish to predict its likely sale price. As another example, we know the cost and condition of a trading card for sale, and we wish to predict a likely profit in buying it and then reselling it. As yet another example, we have a picture with some missing pixels – perhaps there was text covering them, and we want to replace it – and we want to fill in the missing values. As a final example, you can think of classification as a special case of regression, where we want to predict either $+1$ or -1. Predicting values is very useful, and so there are many examples like this.

Second, we might want to compare trends in data. Doing so could make it clear what is really happening. Here is an example from Efron ("Computer-Intensive methods in statistical regression", B. Efron, SIAM Review, 1988). Table 1 shows some data from medical devices, which sit in the body and release a hormone. The data shows the amount of hormone currently in a device after it has spent some time in service, and the time the device spent in service. The data describes devices from three production lots (A, B, and C). Each device, from each lot, is supposed to have the same behavior. The important question is: Are the lots the same? The amount of hormone changes over time, so we can't just compare the amounts currently in each device. Instead, we need to determine the relationship between time in service and hormone, and see if this relationship is different between batches. We can do so by regressing hormone against time.

Figure 13.1 shows how a regression can help. In this case, we have modelled the amount of hormone in the device as

$$a \times (\text{time in service}) + b$$

for a, b chosen to get the best fit (much more on this point later!). This means we can plot each data point on a scatter plot, together with the best fitting line. This plot allows us to ask whether any particular batch behaves differently from the overall model in any interesting way.

However, it is hard to evaluate the distances between data points and the best fitting line by eye. A sensible alternative is to subtract the amount of hormone predicted by the model from the amount that was measured. Doing so yields a **residual** — the difference between a measurement and a prediction. We can then plot those residuals (Figure 13.2). In this case, the plot suggests that lot A is special — all devices from this lot contain less hormone than our model predicts.

Batch A		Batch B		Batch C	
Amount of Hormone	Time in Service	Amount of Hormone	Time in Service	Amount of Hormone	Time in Service
25.8	99	16.3	376	28.8	119
20.5	152	11.6	385	22.0	188
14.3	293	11.8	402	29.7	115
23.2	155	32.5	29	28.9	88
20.6	196	32.0	76	32.8	58
31.1	53	18.0	296	32.5	49
20.9	184	24.1	151	25.4	150
20.9	171	26.5	177	31.7	107
30.4	52	25.8	209	28.5	125

TABLE 13.1: *A table showing the amount of hormone remaining and the time in service for devices from lot A, lot B and lot C. The numbering is arbitrary (i.e. there's no relationship between device 3 in lot A and device 3 in lot B). We expect that the amount of hormone goes down as the device spends more time in service, so cannot compare batches just by comparing numbers.*

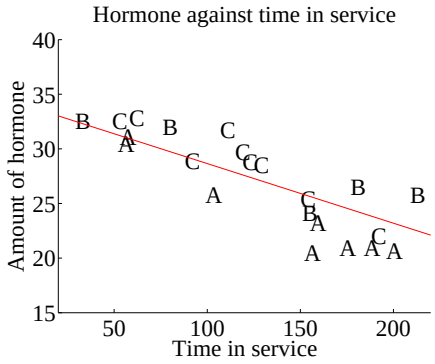

FIGURE 13.1: *A scatter plot of hormone against time for devices from tables 13.1 and 13.1. Notice that there is a pretty clear relationship between time and amount of hormone (the longer the device has been in service the less hormone there is). The issue now is to understand that relationship so that we can tell whether lots A, B and C are the same or different. The best fit line to all the data is shown as well, fitted using the methods of section 13.1.*

13.1 LINEAR REGRESSION AND LEAST SQUARES

Assume we have a dataset consisting of a set of N pairs $(\mathbf{x}_i, y_i)$. We think of y_i as the value of some function evaluated at $\mathbf{x}_i$, with some random component added. This means there might be two data items where the $\mathbf{x}_i$ are the same, and the y_i are different. We refer to the $\mathbf{x}_i$ as **explanatory variables** and the y_i is a **dependent variable**. We want to build a model of the dependence between y and $\mathbf{x}$. This model will be used to predict values of y for new values of $\mathbf{x}$, or

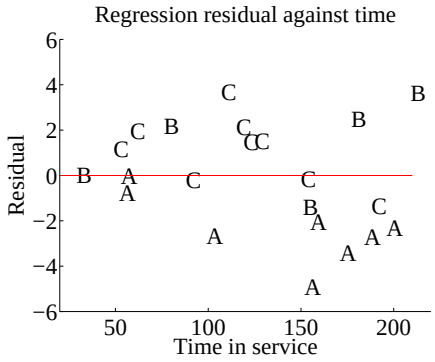

FIGURE 13.2: *This is a scatter plot of residual — the distance between each data point and the best fit line — against time for the devices from tables 13.1 and 13.1. Now you should notice a clear difference; some devices from lots B and C have positive and some negative residuals, but all lot A devices have negative residuals. This means that, when we account for loss of hormone over time, lot A devices still have less hormone in them. This is pretty good evidence that there is a problem with this lot.*

to understand the relationships between the $\mathbf{x}$. The model needs to have some probabilistic component; we do not expect that y is a function of $\mathbf{x}$, and there is likely some error in evaluating y anyhow.

13.1.1 Linear Regression

A good, simple model is to assume that the dependent variable is obtained by evaluating a linear function of the explanatory variables, then adding a zero-mean normal random variable. We can write this model as

$$y = \mathbf{x}^T\beta + \xi$$

where ξ is a zero mean normal random variable with unknown variance (we will be able to estimate this later). In this expression, β is a vector of weights, which we must estimate. When we use this model to predict a value of y for a particular set of explanatory variables $\mathbf{x}^*$, we cannot predict the value that ξ will take. Our best available prediction is the mean value (which is zero).

Example: 13.1 *A linear model fitted to a single explanatory variable*

Assume we fit a linear model to a single explanatory variable. Then the model has the form $y = x\beta + \xi$, where ξ is a zero mean random variable. For any value x^* of the explanatory variable, our best estimate of y is βx^*. In particular, if $x^* = 0$, the model predicts $y = 0$, which is unfortunate. We can draw the model by drawing a line through the origin with slope β in the x, y plane. The y-intercept of this line must be zero.

Example: 13.2 *A linear model with a non-zero y-intercept*

Assume we have a single explanatory variable, which we write u. We can then create a *vector* $\mathbf{x} = [u, 1]^T$ from the explanatory variable. We now fit a linear model to this vector. Then the model has the form $y = \mathbf{x}^T\beta + \xi$, where ξ is a zero mean random variable. For any value $\mathbf{x}^* = [u^*, 1]^T$ of the explanatory variable, our best estimate of y is $(\mathbf{x}^*)^T\beta$, which can be written as $y = \beta_1 u^* + \beta_2$. If $x^* = 0$, the model predicts $y = \beta_2$. We can draw the model by drawing a line through the origin with slope β_1 and y-intercept β_2 in the x, y plane.

The next step is to determine β. Because we have that $P(y|x, \beta)$ is normal, with mean $\mathbf{x}^T\beta$, we can write out the log-likelihood of the data. Write σ^2 for the variance of ξ, which we don't know, but will not need to worry about right now. We have that

$$\begin{aligned}\log \mathcal{L}(\beta) &= \sum_i \log P(y_i|\mathbf{x}_i, \beta) \\ &= \frac{1}{2\sigma^2}\sum_i (y_i - \mathbf{x}_i^T\beta)^2 + \text{term not depending on } \beta\end{aligned}$$

Maximizing the log-likelihood of the data is equivalent to minimizing the negative log-likelihood of the data. Furthermore, the term $\frac{1}{2\sigma^2}$ does not affect the location of the minimum. We must have that β minimizes

$$\sum_i (y_i - \mathbf{x}_i^T\beta)^2.$$

We can write all this more conveniently using vectors and matrices. Write $\mathbf{y}$ for the vector

$$\begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{pmatrix}$$

and $\mathcal{X}$ for the matrix

$$\begin{pmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \ldots \mathbf{x}_n^T \end{pmatrix}.$$

Then we want to minimize

$$(\mathbf{y} - \mathcal{X}\beta)^T(\mathbf{y} - \mathcal{X}\beta)$$

which means that we must have

$$\mathcal{X}^T\mathcal{X}\beta - \mathcal{X}^T\mathbf{y} = 0.$$

For reasonable choices of features, we could expect that $\mathcal{X}^T\mathcal{X}$ — which should strike you as being a lot like a covariance matrix — has full rank. If it does, this equation is easy to solve. If it does not, there is more to do, which we will do below once we have done some examples.

Procedure: 13.1 *Linear Regression with a Normal Model*

We have a dataset containing N pairs $(\mathbf{x}_i, y_i)$. Each x_i is a d-dimensional explanatory vector, and each y_i is a single dependent variable. We assume that each data point conforms to the model

$$y_i = \mathbf{x}_i^T \beta + \xi_i$$

where ξ_i is a normal random variable with mean 0 and unknown variance σ^2. Write $\mathbf{y}$ for the vector

$$\begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{pmatrix}$$

and $\mathcal{X}$ for the matrix

$$\begin{pmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \dots \mathbf{x}_n^T \end{pmatrix}.$$

We estimate $\hat{\beta}$ (the value of β) by solving the linear system

$$\mathcal{X}^T \mathcal{X} \hat{beta} - \mathcal{X}^T \mathbf{y} = 0.$$

The residuals are

$$\mathbf{e} = \mathbf{y} - \mathcal{X}\hat{\beta}.$$

We have that $\mathbf{e}^T \mathbf{1} = 0$. The **mean square error** is given by

$$m = \frac{\mathbf{e}^T \mathbf{e}}{N}.$$

We can estimate σ^2 by

$$\sigma^2 = \frac{\sum_i (y_i - \mathbf{x}_i^T \hat{\beta})^2}{N} = \frac{\mathbf{e}^T \mathbf{e}}{N} = m.$$

Listing 13.1: R code used for the linear regression example of worked example 1

```
setwd('/users/daf/Current/courses/Probcourse/Regression/RCode/EfronDevices');
efd<-read.table('efrontable.txt',header=TRUE)
summary(efd)
# the table has the form
#N1   Ah   Bh   Ch N2   At   Bt   Ct
# now we need to construct a new dataset
#
hor<-stack(efd, select=2:4)
tim<-stack(efd, select=6:8)
foo<-data.frame(time=tim[, c("values")], hormone=hor[, c("values")])
foo.lm<-lm(hormone~time,data=foo)
summary(foo.lm)
#
```

Worked example 13.1 *Simple Linear Regression with R*

Regress the hormone data against time for all the devices in the Efron example.

Solution: This example is mainly used to demonstrate how to regress in R. There is sample code in listing 13.2. The summary in the listing produces a great deal of information (try it). You will see information about the residuals; the smallest is -4.9357, the largest is 3.7323, and the median, first and third quartiles are there too. You will see the intercept (34.167528) and the coefficient of time (-0.057446), as well as a body of statistical information about these coefficients. The R-squared (which we discuss below) is 0.8688. You can get a figure by doing `plot(foo.lm)`, but these figures will not mean anything yet. We discuss some of them below.

13.1.2 Checking Goodness of Fit Qualitatively

It is quite important to know whether the regression is helpful. For example, it seems highly unlikely that regressing the first digit of a telephone number against some numerological score of your name will work that well (or at all). For a dataset with one explanatory variable and one dependent variable, it is quite natural to plot the data on a scatter plot, then plot the model as a line on that scatterplot. Just looking at the picture can be informative (Figure 13.4). But we need more powerful tools, so we can deal with models that are harder to plot.

Assume we have fitted a model $y = \mathbf{x}^T\beta + \xi$ to a dataset, and found the best vector of parameters is $\hat{\beta}$. Then the residual vector is $\mathbf{e} = \mathbf{y} - \mathcal{X}\hat{\beta}$. Inspecting this residual vector will reveal a great deal about a model. We look at qualitative properties first. If the line is horizontal, or close, then the value of the explanatory variable makes very little contribution to the prediction. This suggests that there is no particular relationship between the two. The prediction will be good only if it should take a constant value. Similarly, if the data points lie far from the regression

line, which means that the residual is persistently large compared to the predicted value, then the regression will predict poorly. Figure 13.3 shows a linear regression of age against weight — i.e., this builds a model that tries to predict someone's age from a knowledge of their weight. You won't be surprised that the predictions are poor, which can be confirmed by looking at the figure.

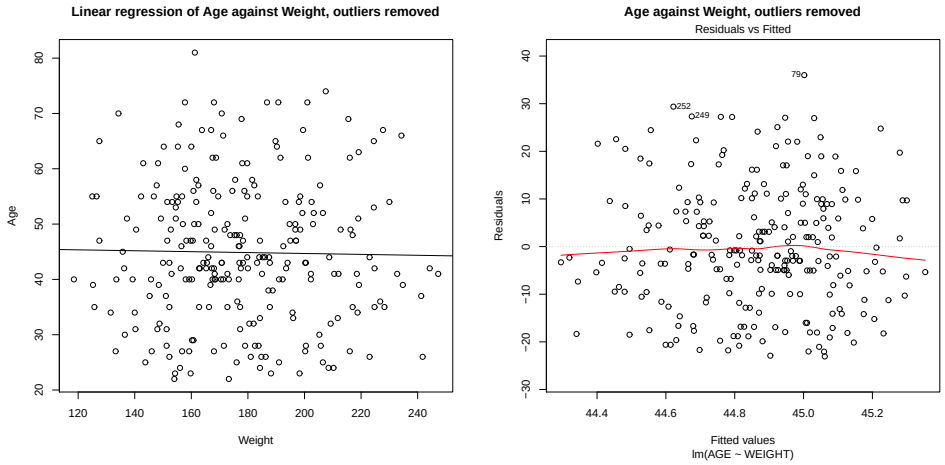

FIGURE 13.3: *On the* **left**, *age regressed against weight for the bodyfat dataset, with four outliers removed. The data are presented as a scatter plot, and the line is the regression line. Notice that the line is nearly horizontal, suggesting that knowing weight does not improve one's prediction of age. Notice also that many data points lie rather far from the line. On the* **right**, *a scatter plot of the residual against the value predicted by the regression. Generally, the residual takes rather large values, suggesting that predictions are unreliable.*

We assumed that $y - \mathbf{x}^T\beta$ was a zero-mean normal random variable with fixed variance. In turn, this means that the value of the residual vector should not depend on the corresponding y-value. If it does, this is good evidence that there is a problem. Looking at a scatter plot of $\mathbf{e}$ against $\mathbf{y}$ will often reveal trouble in a regression (Figure 13.4).

In the case of Figure 13.4, the trouble is caused by a few outlying data points severely affecting the regression. We will discuss how to identify and deal with such points in Section 1. Once they have been removed, the regression improves markedly (Figure 13.5).

Figure **??** shows another example, based on the idea of word frequencies. Some words are used very often in text; most are used seldom. The dataset for this figure consists of counts of the number of time a word occurred for the 100 most common words in Shakespeare's printed works. It was originally collected from a concordance, and has been used to attack a variety of interesting questions, including an attempt to assess how many words Shakespeare knew. This is hard, because he likely knew many words that he didn't use in his works, so one can't just count. If you look at the plot of Figure 13.11, you can see that a linear regression

FIGURE 13.4: *On the* **left**, *weight regressed against height for the bodyfat dataset. The data are presented as a scatter plot, and the line is the regression line. Notice the line doesn't describe the data particularly well, because it has been strongly affected by a few data points. On the* **right**, *a scatter plot of the residual against the value predicted by the regression. Large values at some points are another sign of trouble in this regression.*

of count (the number of times a word is used) against rank (how common a word is, 1-100) is not really useful (Figure 13.11). It doesn't model the very high frequencies with which common words are used. You can see this effect in the scatter plot of residual against dependent variable in Figure 13.11 — the residual depends rather strongly on the dependent variable.

13.1.3 Evaluating Goodness of Fit

There is an important quantitative measure of how good a regression is. The dependent variable has some variance (unless it's a constant, which makes prediction easy). If our model is of any use, it should explain some aspects of the value of the dependent variable. This means that the variance of the residual should be smaller than the variance of the dependent variable. If the model made perfect predictions, then the variance of the residual should be zero.

We can formalize all this in a relatively straightforward way. We will ensure that $\mathcal{X}$ always has a column of ones in it, so that the regression can have a non-zero y-intercept. We now fit a model

$$\mathbf{y} = \mathcal{X}\beta + \mathbf{e}$$

(where $\mathbf{e}$ is the vector of residual values) by choosing β such that $\mathbf{e}^T\mathbf{e}$ is minimized. Then we get some useful technical results.

FIGURE 13.5: *On the* **left**, *weight regressed against height for the bodyfat dataset. I have now removed four data points, which were the most likely to have been outliers. The data are presented as a scatter plot, and the line is the regression line. The line describes the data rather well in comparison with figure 13.4 (notice the axes are different). On the* **right**, *a scatter plot of the residual against the value predicted by the regression. Notice that the mean squared residual is about the same for different values of the prediction; this is consistent with our model, and suggests the regression will yield good predictions.*

Useful Facts: 13.1 *Regression*

We write $\mathbf{y} = \mathcal{X}\beta + \mathbf{e}$, where $\mathbf{e}$ is the residual. Assume $\mathcal{X}$ has a column of ones, and β is chosen to minimize $\mathbf{e}^T\mathbf{e}$. Then we have

1. $\mathbf{e}^T\mathcal{X} = \mathbf{0}$, i.e. that $\mathbf{e}$ is orthogonal to any column of $\mathcal{X}$. This is because, if $\mathbf{e}$ is not orthogonal to some column of $\mathbf{e}$, we can increase or decrease the β term corresponding to that column to make the error smaller. Another way to see this is to notice that *beta* is chosen to minimize $\mathbf{e}^T\mathbf{e}$, which is $(\mathbf{y}-\mathcal{X}\beta)^T(\mathbf{y}-\mathcal{X}\beta)$. Now because this is a minimum, the gradient with respect to β is zero, so $(\mathbf{y} - \mathcal{X}\beta)^T(-\mathcal{X}) = -\mathbf{e}^T\mathcal{X} = 0$.
2. $\mathbf{e}^T\mathbf{1} = 0$ (recall that $\mathcal{X}$ has a column of all ones, and apply the previous result).
3. $\mathbf{1}^T(\mathbf{y} - \mathcal{X}\beta) = 0$ (same as previous result).
4. $\mathbf{e}^T\mathcal{X}\beta = 0$ (first result means that this is true).

Now $\mathbf{y}$ is a one dimensional dataset arranged into a vector, so we can compute

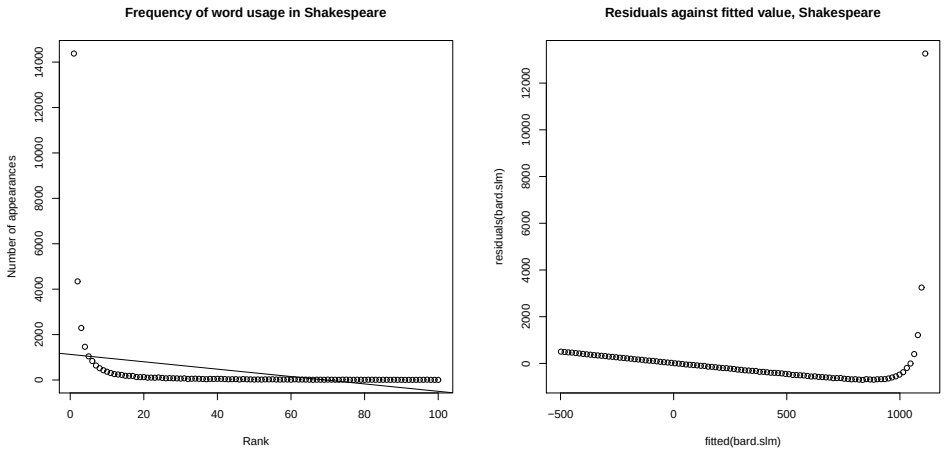

FIGURE 13.6: *On the* **left**, *word count plotted against rank for the 100 most common words in Shakespeare, using the data of ****. I show a regression line too. This is a poor fit by eye, confirmed by the plot of the residuals on the* **right**, *which shows the residuals plotted against dependent variable for this regression. Notice that the residual value depends very strongly on the value of the dependent variable.*

$\mathsf{mean}\,(\{y\})$ and $\mathsf{var}[y]$. Similarly, $\mathcal{X}\beta$ is a one dimensional dataset arranged into a vector (its elements are $\mathbf{x}_i^T\beta$), as is $\mathbf{e}$, so we know the meaning of mean and variance for each. We have a particularly important result:

$$\mathsf{var}[y] = \mathsf{var}[\mathcal{X}\beta] + \mathsf{var}[e].$$

This is quite easy to show, with a little more notation. Write $\overline{\mathbf{y}} = (1/N)(\mathbf{1}^T\mathbf{y})\mathbf{1}$ for the vector whose entries are all $\mathsf{mean}\,(\{y\})$; similarly for $\overline{e}$ and for $\overline{\mathcal{X}\beta}$. We have

$$\mathsf{var}[y] = (1/N)(\mathbf{y} - \overline{\mathbf{y}})^T(\mathbf{y} - \overline{\mathbf{y}})$$

and so on for $\mathsf{var}[e_i]$, etc. Notice from the facts that $\overline{\mathbf{y}} = \overline{\mathcal{X}\beta}$. Now

$$\begin{aligned}
\mathsf{var}[y] &= (1/N)\left(\left[\mathcal{X}\beta - \overline{\mathcal{X}\beta}\right] + \left[\mathbf{e} - \overline{\mathbf{e}}\right]\right)^T\left(\left[\mathcal{X}\beta - \overline{\mathcal{X}\beta}\right] + \left[\mathbf{e} - \overline{\mathbf{e}}\right]\right) \\
&= (1/N)\left(\left[\mathcal{X}\beta - \overline{\mathcal{X}\beta}\right]^T\left[\mathcal{X}\beta - \overline{\mathcal{X}\beta}\right] + 2\left[\mathbf{e} - \overline{\mathbf{e}}\right]^T\left[\mathcal{X}\beta - \overline{\mathcal{X}\beta}\right] + \left[\mathbf{e} - \overline{\mathbf{e}}\right]^T\left[\mathbf{e} - \overline{\mathbf{e}}\right]\right) \\
&= (1/N)\left(\left[\mathcal{X}\beta - \overline{\mathcal{X}\beta}\right]^T\left[\mathcal{X}\beta - \overline{\mathcal{X}\beta}\right] + \left[\mathbf{e} - \overline{\mathbf{e}}\right]^T\left[\mathbf{e} - \overline{\mathbf{e}}\right]\right) \\
&\quad \text{because } \overline{\mathbf{e}} = 0 \text{ and } \mathbf{e}^T\mathcal{X}\beta = 0 \text{ and } \mathbf{e}^T\mathbf{1} = 0 \\
&= \mathsf{var}[\mathcal{X}\beta] + \mathsf{var}[e].
\end{aligned}$$

This is extremely important, because us allows us to think about a regression as explaining variance in $\mathbf{y}$. As we are better at explaining $\mathbf{y}$, $\mathsf{var}[e]$ goes down. In turn, a natural measure of the goodness of a regression is what percentage of the variance of $\mathbf{y}$ it explains. This is known as R^2 (the r-squared measure). We have

$$R^2 = \frac{\mathsf{var}\left[\mathbf{x}_i^T\beta\right]}{\mathsf{var}[y_i]}$$

which gives some sense of how well the regression explains the training data.

Good predictions result in high values of R^2, and a perfect model will have $R^2 = 1$ (which doesn't usually happen). For example, the regression of figure 13.1 has an R^2 value of 0.87; the regression of weight against all explanatory variables (Figure 13.8) has an R^2 value of 0.99; and the regression of Boston house prices against all explanatory variables (Figure 13.8) has an R^2 value of 0.72. Similarly, poor predictions result in low values of R^2. For example, the regression of weight on height of Figure 13.4 has an R^2 of 0.095; the regression of age on weight of Figure 13.3 has an R^2 of 0.00030 (because knowing weight tells one essentially nothing about age or height). Removing the outliers from the bodyfat dataset, to get the regression of weight on height of Figure 13.5, improves the R^2 to 0.29.

13.1.4 Linear Regression: Examples

For the data of tables 13.1 and 13.1, y is the hormone remaining in the device and $\mathbf{x} = (\text{time}, 1)$. This gives us a model of the hormone in the device as

$$y = \beta_1 \text{time} + \beta_2$$

which should look like the line in figure **??**. We get $\beta = (-0.0574, 34.2)$. Now we can ask whether some lots of device behave differently than others. One way to address this is to consider the **residual**,

$$y_i - \mathbf{x}_i^T \beta$$

which is the difference between the observed value and what the model predicts. Look at figure 13.2, which shows this residual plotted against the time. Notice that, for batch A, the model always over predicts, whereas batches B and C seem about the same; this suggests that there is something different about A — any effects caused by the hormone being taken up from the device have been accounted for by our model, and the residual shows the effects that are left.

13.2 PRODUCING GOOD LINEAR REGRESSIONS

Outlying data points can significantly weaken the usefulness of a regression. We need to identify data points that might be a problem, and then resolve how to deal with them. One possibility is that they are true outliers — someone recorded a data item wrong, or they represent an effect that just doesn't occur all that often. Another is that they are important data, and our linear model may not be good enough. If the data points really are outliers, we can ignore them; if they aren't, we may be able to improve the regression by transforming features or by finding a new explanatory variable.

13.2.1 Problem Data Points

We are solving for the β that minimizes $\sum_i (y_i - \mathbf{x}_i^T \beta)^2$, equivalently for the β that produces the smallest value of $\sum_i e_i^2$. This means that residuals with large value can have a very strong influence on the outcome — we are squaring that large value. Generally, many residuals of medium size will have a smaller cost that one large residual and the rest tiny. In turn, some data points can have an excessive

influence on the choice of β. This creates a problem, because data points that are clearly wrong (sometimes called **outliers**) can also have the highest influence on the outcome of the regression. Compare Figure 13.4 and Figure 13.5 to see this effect for a simple case.

When we have only one explanatory variable, there's an easy method to spot problem data points. One produces a scatter plot and a regression line, and the difficulty is usually obvious. In particularly tricky cases, printing the plot and using a perspex ruler to draw a line by eye can help. When there are more, we need more powerful visualization tools.

There are two tools that are simple and effective. One method deletes the i'th point, computes the regression for the reduced data set, then compares the true value of *every other point* to the predictions made by the dataset with the i'th point deleted. The score for the comparison is called **Cook's distance**. If a point has a large value of Cook's distance, then it has a strong influence on the regression and might well be an outlier. Typically, one computes Cook's distance for each point, and takes a closer look at any point with a large value. This procedure is described in more detail in procedure 2

Procedure: 13.2 *Computing Cook's distance*

We have a dataset containing N pairs $(\mathbf{x}_i, y_i)$. Each x_i is a d-dimensional explanatory vector, and each y_i is a single dependent variable. Write $\hat{\beta}$ for the coefficients of a linear regression (see procedure 1), and $\hat{\beta}_{\hat{i}}$ for the coefficients of the linear regression computed by omitting the i'th data point, and m for the mean square error. The Cook's distance of the i'th data point is

$$\frac{\sum_j (\mathbf{x}_j^T \hat{\beta} - \mathbf{x}_j^T \hat{\beta}_{\hat{i}})}{dm}.$$

An alternative method identifies data points that have few nearby points by computing **leverage**. Write $\hat{\beta}$ for the estimated value of β, and $\mathbf{y}_p = \mathcal{X}\hat{\beta}$ for the predicted y values. Then we have

$$\hat{\beta} = \left(\mathcal{X}^T\mathcal{X}\right)^{-1}\left(\mathcal{X}^T\mathbf{y}\right)$$

so that

$$\mathbf{y}_p = \left(\mathcal{X}\left(\mathcal{X}^T\mathcal{X}\right)^{-1}\mathcal{X}^T\right)\mathbf{y}.$$

The matrix $\left(\mathcal{X}\left(\mathcal{X}^T\mathcal{X}\right)^{-1}\mathcal{X}^T\right)$ is sometimes called the **hat matrix**, and is written $\mathcal{H}$. It gives the predicted values as a linear function of the observed values. The leverage of the i'th point is the i'th diagonal element, h_{ii}, of the hat matrix $\mathcal{H}$. It is straightforward to show that the eigenvalues of a hat matrix can be only 1 or 0, and that it is symmetric (exercises). It is also straightforward to show that, for the

i'th row of the hat matrix, the sum of squares is less than one, that is

$$\sum_j h_{ij}^2 \leq 1.$$

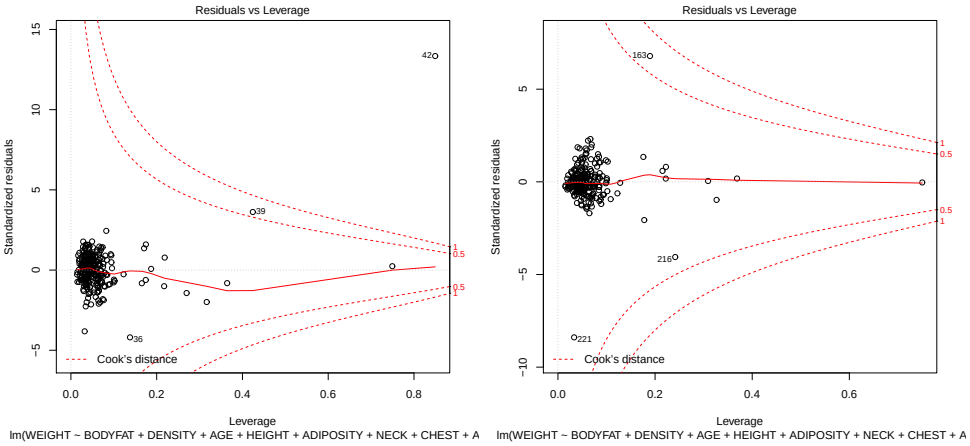

FIGURE 13.7: *On the* **left**, *residuals plotted against leverage for a regression of weight against all other measurements for the bodyfat dataset. I did* not *remove the outliers. The contours on the plot are contours of Cook distance; I have overlaid arrows showing points with suspiciously large Cook distance. Notice also that several points have high leverage, without having a large residual value. These points may or may not present problems. On the* **right**, *the same plot for this dataset with points 36, 39, 41 and 42 removed (these are the points I have been removing for each such plot). Notice that another point now has high Cook distance, but mostly the residual is much smaller.*

You can interpret each row of the hat matrix as a set of mixing weights that mix the observations to produce the prediction for a particular data point. But these weights have limited size, so that if h_{ii} is large, then the other terms must be small. In particular, if h_{ii} is large, then other data items make relatively little contribution to the prediction *at that point*. This could mean that the point is an outlier, or that there are few examples nearby. In either case, it is reasonable to worry about the data point. One might try and get more data, or reduce the number of explanatory variables.

13.2.2 Explanatory variables

When there is more than one explanatory variable, it is difficult to plot the regression. Instead, one can get an idea of the usefulness of the regression by plotting the residual against the predicted value. Figure 13.9 show these plots for the bodyfat dataset, and for a regression of boston house prices against a variety of explanatory variables (the data set is quite well known; you can find it at the

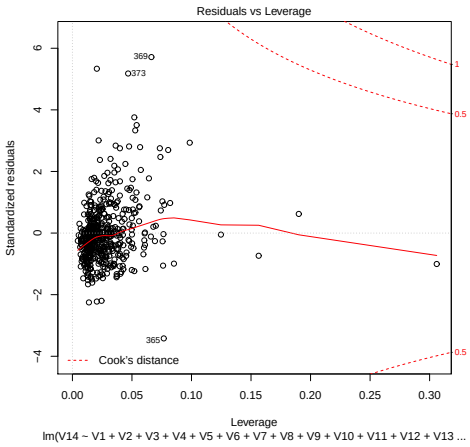

FIGURE 13.8: *Residuals plotted against leverage for a regression of weight against all other measurements for the Boston housing. The contours on the plot are contours of Cook distance; No points have suspiciously large Cook distance, though the software package has identified points most likely to present problems. Notice also that several points have high leverage, without having a large residual value. These points may or may not present problems. The residual plot of Figure 13.9 suggests that this regression will make poor predictions, but the problem does not seem to be outlying points.*

UCI repository, `http://archive.ics.uci.edu/ml/datasets/Housing`). In these plots, you should notice there is some structure — the residuals seem to depend on the predicted value. This suggests that we could improve the regression either by supplying another explanatory variable, or by transforming the variables we have. Both methods are useful.

Sometimes the data isn't in a form that leads to a good linear regression. In this case, transforming explanatory variables, the dependent variable, or both can lead to big improvements. Figure **??** shows one example, based on the idea of word frequencies. Some words are used very often in text; most are used seldom. The dataset for this figure consists of counts of the number of time a word occurred for the 100 most common words in Shakespeare's printed works. It was originally collected from a concordance, and has been used to attack a variety of interesting questions, including an attempt to assess how many words Shakespeare knew. This is hard, because he likely knew many words that he didn't use in his works, so one can't just count.

Notice that a linear regression of count (the number of times a word is used) against rank (how common a word is, 1-100) is not really useful. It doesn't model the very high frequencies with which common words are used. However, if we regress log-count against log-rank, we get a very good fit indeed. This suggests that Shakespeare's word usage (at least for the 100 most common words) is consistent with **Zipf's law**. This gives the relation between frequency f and rank r for a

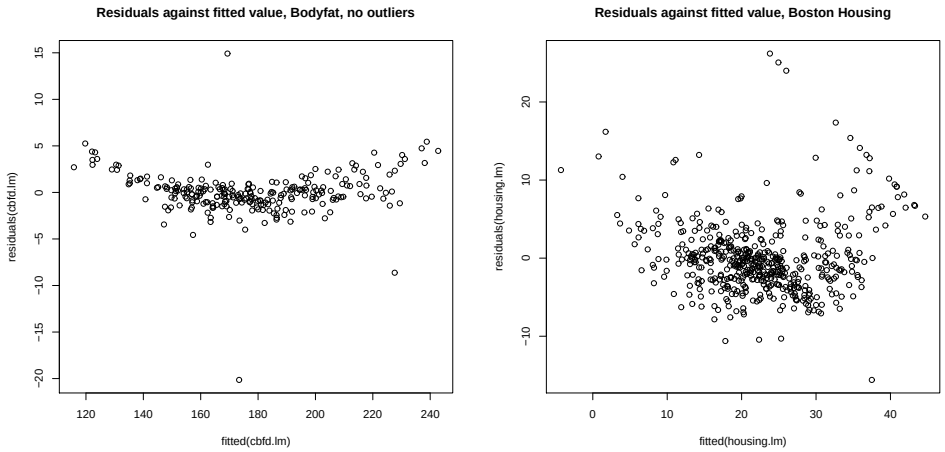

FIGURE 13.9: *On the* **left**, *residuals plotted against predicted value for a regression of weight against all other measurements for the bodyfat dataset, with four outliers removed. This regression is quite successful (small residuals). Notice that the variance of the residuals changes somewhat as the predicted value increases, giving a banana shape to the data. This suggests that we are missing an explanatory variable, or that a non-linear transformation of the explanatory variables might be helpful. On the* **right**, *a scatter plot of the residual against the value predicted by the regression for the price of a house in Boston regressed against a variety of explanatory variables. Generally, the residual takes rather large values. At small values of the prediction, the residual is somewhat larger. This suggests that we are missing an explanatory variable, or that a non-linear transformation of the explanatory variables might be helpful. There is a curious linear structure in the top right corner, which might have been caused by the dependent variable being thresholded.*

word as

$$f \propto \frac{1}{r}^s$$

where s is a constant characterizing the distribution. Our linear regression suggests that s is approximately 1.67 for this data.

In some cases, the natural logic of the problem will suggest variable transformations that improve regression performance. For example, one could argue that humans have approximately the same density, and so that weight should scale as the cube of height; in turn, this suggests that one regress weight against the cube root of height. Generally, shorter people tend not to be scaled versions of taller people, so the cube root might be too aggressive. For example, the body mass index divides weight by the square (rather than the cube) of height. An appropriate transformation has attracted a great deal of interest over the last century and a half, perhaps because there are many confounding variables. For example, the body mass index tends to regard muscular individuals as obese.

In other cases, one might need to try different transformations or groupings of

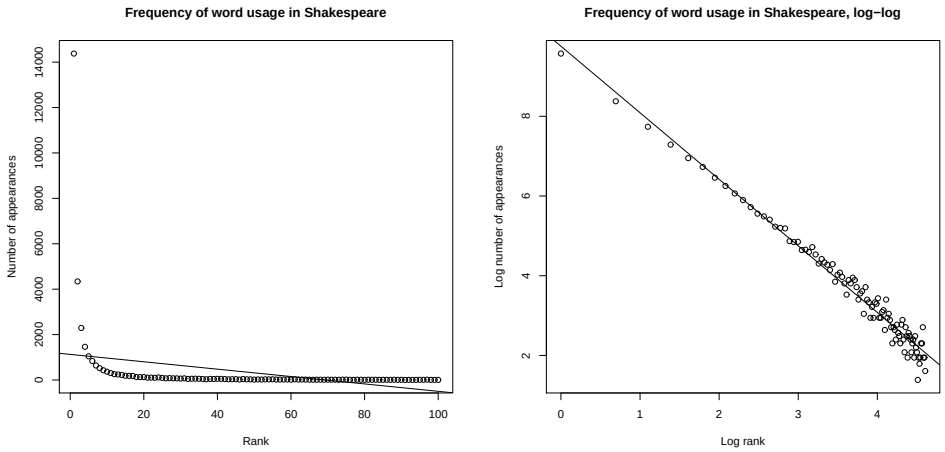

FIGURE 13.10: *On the* **left**, *word count plotted against rank for the 100 most common words in Shakespeare, using the data of ****. I show a regression line too. This is a poor fit by eye, confirmed by Figure* **??**. *On the* **right**, *log word count plotted against log rank for the 100 most common words in Shakespeare, using the data of ****. The regression line is very close to the data, confirmed by Figure* **??**.

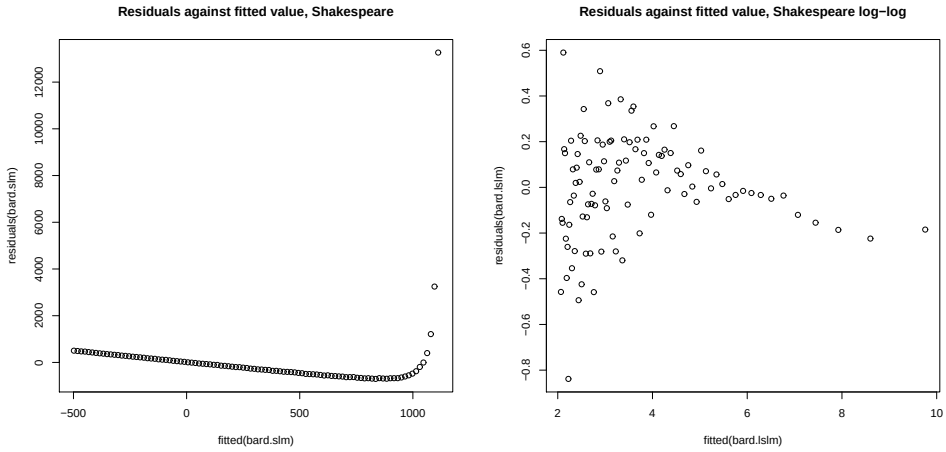

FIGURE 13.11: *On the* **left**, *residuals of word count regressed against rank for the 100 most common words in Shakespeare, plotted against the predicted value. Notice that some large predictions have enormous residuals. This plot suggests serious problems with the regression, which you can confirm by looking at Figure 13.11. On the* **right**, *residuals of log word count regressed against log rank for the 100 most common words in Shakespeare, plotted against the predicted value. Notice that residual variance is about the same at each value, a good sign.*

explanatory variables. There is a method for transforming the dependent variable, the **Box-Cox transformation**, that can search for a transformation that improves the regression. This is a one-parameter family of transformations, with parameter λ. We define the Box-Cox transformation of the dependent variable to be

$$y_i^{(bc)} = \begin{cases} \frac{y_i^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0 \\ \log y_i & \text{if } \lambda = 0 \end{cases} .$$

It turns out to be straightforward to estimate a good value of λ using maximum likelihood. One searches for a value of λ that makes residuals look most like a normal distribution. Statistical software will do it for you; the exercises sketch out the method. This transformation can produce significant improvements in a regression. Figure **??** shows a small but acceptable improvement in regression of weight against height using the Box-Cox transformation (residuals in Figure 13.13). Figure 13.14 shows an improvement in the regression of weight against all variables for the bodyfat dataset obtained using the Box-Cox transformation. Figure 13.15 shows an improvement in the regression of Boston house price against all variables obtained using the Box-Cox transformation.

Worked example 13.2 *BoxCox with R*

Regress weight against height using the Box-Cox transformation

Solution: This example is mainly used to demonstrate how easy it is to do Box-Cox in R. There is sample code in listing **??**. The line `boxcox(...)` will produce a plot on your screen - you look for the location of the peak, and that's the parameter value to use. In this case, the peak is close to zero, so we should work with the log of the dependent variable. Notice how neat `abline` is — you pass it a linear regression, and it puts a line on a plot.

13.3 WHICH VARIABLES ARE IMPORTANT?

13.3.1 Regularizing Linear Regressions

One occasionally important difficulty is that the explanatory variables might be significantly correlated. If they are, then it will generally be easy to predict one explanatory variable from another. This means that $\mathcal{X}^T\mathcal{X}$ may have some very small eigenvalues (because there is a vector $\mathbf{u}$ so that $\mathcal{X}\mathbf{u}$ is small; this means that $\mathbf{u}^T\mathcal{X}^T\mathcal{X}\mathbf{u}$ must be small).

These small eigenvalues lead to bad predictions. If $\mathcal{X}^T\mathcal{X}$ has a small eigenvalue, then there is some vector $\mathbf{v}$ such that $\mathcal{X}^T\mathcal{X}\mathbf{v}$ is small, or, equivalently, that the matrix can turn large vectors into small ones; but that means that $(\mathcal{X}^T\mathcal{X})^{-1}$ will turn some small vectors into big ones. In turn, this means that small errors in $\mathbf{y}$ — which are likely inevitable — will result in big errors in β. This could cause trouble in two ways. If we are looking at β to tell which explanatory variables are important, then large errors in β will be a problem. And if we are trying to predict

Listing 13.2: R code used for the linear regression example of worked example 2

```
setwd('/users/daf/Current/courses/Probcourse/Regression/RCode/HeightWeight');
#install.packages('gdata')
library(gdata)
#cement slag ash water super coarse fine age strength
bfd<-read.xls('BodyFat.xls')
summary(bfd)
#IDNO BODYFAT DENSITY AGE WEIGHT HEIGHT ADIPOSITY NECK CHEST ABDOMEN
HIP THIGH KNEE ANKLE BICEPS FOREARM WRIST X
sbfd.lm<-lm(WEIGHT~HEIGHT,data=bfd)
plot(bfd[,c("HEIGHT")], bfd[, c("WEIGHT")],
     main="Linear regression of Weight against Height", xlab="Height", ylab="Weight")
abline(sbfd.lm)
#36, 39 41 and 42 are the bad ones
cbfd<-bfd[-c(36, 39, 41,42),]
scbfd<-lm(WEIGHT~HEIGHT,data=cbfd)
plot(cbfd[,c("HEIGHT")], cbfd[, c("WEIGHT")],
     main="Linear regression of Weight against Height, outliers removed', xlab="Height", ylab=
abline(scbfd)

library(MASS)
cbfd.lm<-lm(WEIGHT~HEIGHT,data=cbfd)
boxcox(cbfd.lm, plotit=TRUE)
# suggests lambda=0 is best, strongly
cbfd.llm<-lm(log(WEIGHT)~HEIGHT,data=cbfd)
plot(cbfd[,c("HEIGHT")], log(cbfd[, c("WEIGHT")]),
     main="Linear regression of log Weight against Height, outliers removed", xlab="Height", y
abline(cbfd.llm)
```

new y values, we expect that errors in β turn into errors in prediction.

An important and useful way to suppress these errors is to regularize, using the same trick we saw in the case of classification. Instead of choosing β to minimize

$$\sum_i (y_i - \mathbf{x}_i^T\beta)^2 = (\mathbf{y} - \mathcal{X}\beta)^T(\mathbf{y} - \mathcal{X}\beta)$$

we minimize

$$\sum_i (y_i - \mathbf{x}_i^T\beta)^2 + \lambda\beta^T\beta = (\mathbf{y} - \mathcal{X}\beta)^T(\mathbf{y} - \mathcal{X}\beta) + \lambda\beta^T\beta$$

where $\lambda > 0$ is a constant. We choose λ in the same way we used for classification; split the training set into a training piece and a validation piece, train for different values of λ, and test the resulting regressions on the validation piece. We choose the λ that yields the smallest validation error. Notice we could use multiple splits, and average over the splits.

This helps, because to solve for β we must solve the equation

$$(\mathcal{X}^T\mathcal{X} + \lambda\mathcal{I})\beta = \mathcal{X}^T\mathbf{y}$$

(obtained by differentiating with respect to β and setting to zero) and the smallest eigenvalue of the matrix $(\mathcal{X}^T\mathcal{X} + \lambda\mathcal{I})$ will be at least λ.

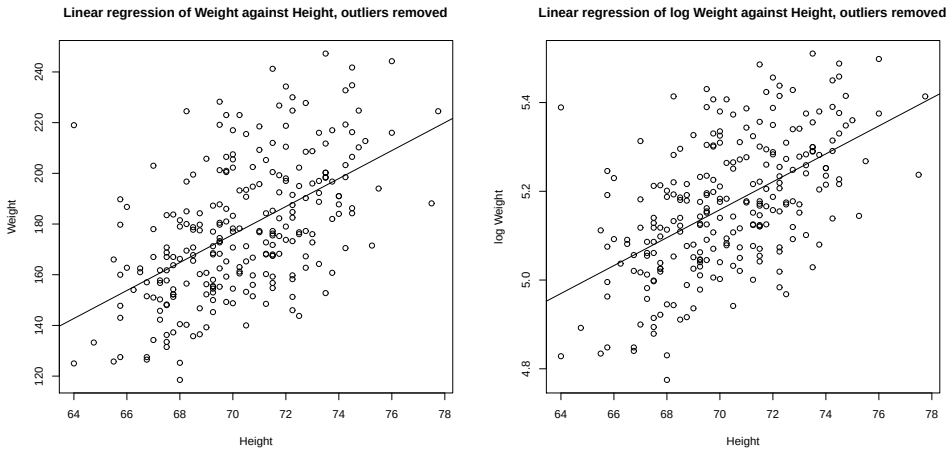

FIGURE 13.12: *On the* **left**, *a linear regression of weight against height for the bodyfat dataset. This is just a copy of the regression of Figure 1, for reference. The* R^2 *is 0.29. The Box-Cox method predicts that one should use* $\lambda = 0$ *(equivalently, predict* $\log$ *weight from height). On the* **right**, *a linear regression of log weight against height. The regression is slightly better (*R^2 *is 0.3). Figure 1 compares the residuals for these cases.*

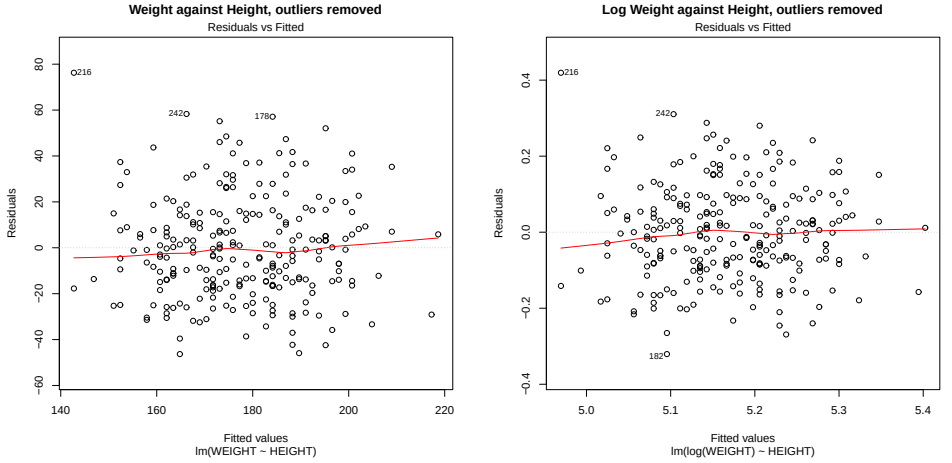

FIGURE 13.13: *On the* **left**, *the residuals against predicted value for the linear regression of weight against height for the bodyfat dataset. This is just a copy of the regression of Figure 1, for reference. The* R^2 *is 0.29. The Box-Cox method predicts that one should use* $\lambda = 0$ *(equivalently, predict* $\log$ *weight from height). On the* **right**, *a linear regression of the log of weight against height. The regression is slightly better (*R^2 *is 0.3).*

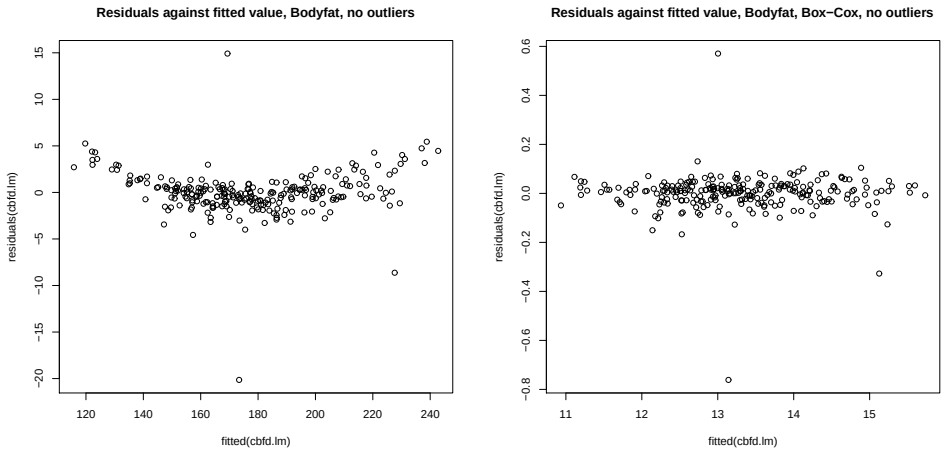

FIGURE 13.14: *On the* **left**, *residuals plotted against predicted value for a regression of weight against all other measurements for the bodyfat dataset, with four outliers removed. Notice that the variance of the residuals changes somewhat as the predicted value increases, giving a banana shape to the data. This suggests that a non-linear transformation of the explanatory variables might be helpful. The Box-Cox procedure suggests* $\lambda = 0.5$, *so we regress* $\sqrt{weight}$ *against the explanatory variables. This yields the plot on the* **right**. *Notice how the banana shape has gone, and the residuals have about the same distribution for each predicted value.*

13.3.2 Which Variables should Contribute? The LASSO

13.4 WHAT YOU SHOULD REMEMBER - NEED SOMETHING

FIGURE 13.15: *On the* **left**, *residuals plotted against predicted value for a regression of Boston house prices against all other measurements (compare Figure 1). The Box-Cox procedure suggests* $\lambda = 0.1$, *so we regress* $weight^{0.1}$ *against the explanatory variables. This yields the plot on the* **right**. *Notice how the residuals have about the same distribution for each predicted value, but the linear structure hasn't gone.*

CHAPTER 14

Learning to Classify

A **classifier** is a procedure that accepts a set of features and produces a class label for them. There could be two, or many, classes, though it is usual to produce multi-class classifiers out of two-class classifiers. Classifiers are immensely useful, and find wide application, because many problems are naturally decision problems. For example, if you wish to determine whether to place an advert on a web-page or not, you would use a classifier (i.e. look at the page, and say yes or no according to some rule). As another example, if you have a program that you found for free on the web, you would use a classifier to decide whether it was safe to run it (i.e. look at the program, and say yes or no according to some rule). As yet another example, you can think of doctors as extremely complex multi-class classifiers.

Classifiers are built by taking a set of labeled examples and using them to come up with a rule that assigns a label to any new example. In the general problem, we have a training dataset $(\mathbf{x}_i, y_i)$; each of the **feature vectors** $\mathbf{x}_i$ consists of measurements of the properties of different types of object, and the y_i are labels giving the type of the object that generated the example.

14.1 CLASSIFICATION, ERROR, AND LOSS

You should think of a classifier as a rule, though it might not be implemented that way. We pass in a feature vector, and the rule returns a class label. We know the relative costs of mislabeling each class and must come up with a rule that can take any plausible $\mathbf{x}$ and assign a class to it, in such a way that the expected mislabeling cost is as small as possible, or at least tolerable. For most of this chapter, we will assume that there are two classes, labeled 1 and -1. Section 14.5.2 shows methods for building multi-class classifiers from two-class classifiers.

14.1.1 Using Loss to Determine Decisions

The choice of classification rule must depend on the cost of making a mistake. A two-class classifier can make two kinds of mistake. A **false positive** occurs when a negative example is classified positive; a **false negative** occurs when a positive example is classified negative. For example, pretend there is only one disease; then doctors would be classifiers, deciding whether a patient had it or not. If this disease is dangerous, but is safely and easily treated, then false negatives are expensive errors, but false positives are cheap. Similarly, if it is not dangerous, but the treatment is difficult and unpleasant, then false positives are expensive errors and false negatives are cheap.

14.1.2 Training Error, Test Error, and Overfitting

It can be quite difficult to know a good loss function, but one can usually come up with a plausible model. If we knew the posterior probabilities, building a classifier would be straightforward. Usually we don't, and must build a model from data. This model could be a model of the posterior probabilities, or an estimate of the decision boundaries. In either case, we have only the training data to build it with. **Training error** is the error a model makes on the training data set.

Generally, we will try to make this training error small. However, what we really want to minimize is the **test error**, the error the classifier makes on test data. We cannot minimize this error directly, because we don't know the test set (if we did, special procedures in training apply). However, classifiers that have small training error might not have small test error. One example of this problem is the (silly) classifier that takes any data point and, if it is the same as a point in the training set, emits the class of that point and otherwise chooses randomly between the classes. This classifier has been learned from data, and has a zero error rate on the training dataset; it is likely to be unhelpful on any other dataset, however.

The phenomenon that causes test error to be worse than training error is sometimes called **overfitting** (other names include **selection bias**, because the training data has been selected and so isn't exactly like the test data, and **generalizing badly**, because the classifier fails to generalize). It occurs because the classifier has been trained to perform well *on the training dataset*. The training dataset is not the same as the test dataset. First, it is quite likely smaller. Second, it might be biased through a variety of accidents. This means that small training error may have to do with quirks of the training dataset that don't occur in other sets of examples. It is quite possible that, in this case, the test error will be larger than the training error. Generally, we expect classifiers to perform somewhat better on the training set than on the test set. Overfitting can result in a substantial difference between performance on the training set and performance on the test set. One consequence of overfitting is that classifiers should always be evaluated on test data. Doing this creates other problems, which we discuss in Section 14.1.3.

A procedure called **regularization** attaches a penalty term to the training error to get a better estimate of the test error. This penalty term could take a variety of different forms, depending on the requirements of the application. Section 1 describes regularization in further detail.

14.1.3 Error Rate and Cross-Validation

There are a variety of methods to describe the performance of a classifier. Natural, straightforward choices are to report the **error rate**, the percentage of classification attempts on a test set that result in the wrong answer. This presents an important difficulty. We cannot estimate the error rate of the classifier using training data, because the classifier has been trained to do well on that data, which will mean our error rate estimate will be an underestimate. An alternative is to separate out some training data to form a **validation set**, then train the classifier on the rest of the data, and evaluate on the validation set. This has the difficulty that the classifier will not be the best estimate possible, because we have left out some training data

when we trained it. This issue can become a significant nuisance when we are trying to tell which of a set of classifiers to use—did the classifier perform poorly on validation data because it is not suited to the problem representation or because it was trained on too little data?

We can resolve this problem with **cross-validation**, which involves repeatedly: splitting data into training and validation sets uniformly and at random, training a classifier on the training set, evaluating it on the validation set, and then averaging the error over all splits. This allows an estimate of the likely future performance of a classifier, at the expense of substantial computation.

Choose some class of subsets of the training set,
for example, singletons.

For each element of that class, construct a classifier by
omitting that element in training, and compute the
mean number of classification errors on the omitted subset.

Average these errors over the class of subsets to estimate
the risk of using the classifier trained on the entire training
dataset.

Algorithm 14.1: *Cross-Validation*

The most usual form of this algorithm involves omitting single items from the dataset and is known as **leave-one-out cross-validation**. Errors are usually estimated by simply averaging over the class, but more sophisticated estimates are available. We do not justify this tool mathematically; however, it is worth noticing that leave-one-out cross-validation, in some sense, looks at the sensitivity of the classifier to a small change in the training set. If a classifier performs well under this test, then large subsets of the dataset look similar to one another, which suggests that a representation of the relevant probabilities derived from the dataset might be quite good.

14.2 LINEAR CLASSIFIERS

Assume we have a set of N example points $\mathbf{x}_i$ that belong to two classes, which we indicate by 1 and -1. These points come with their class labels, which we write as y_i; thus, our dataset can be written as

$$\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}.$$

We wish to the sign of y for any point $\mathbf{x}$; this rule is our classifier. Write $y_i^{(p)}(\mathbf{x})$ for the predicted value of y for a given value of $\mathbf{x}$. assume we have a set of N example points $\mathbf{x}_i$ that belong to two classes, which we indicate by 1 and -1. These points come with their class labels, which we write as y_i; thus, our dataset can be written as

$$\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}.$$

We seek a rule that predicts the sign of y for any point $\mathbf{x}$; this rule is our classifier. We will use a linear rule, so that for a new data item $\mathbf{x}$, we will predict

$$\text{sign}\,((\mathbf{w} \cdot \mathbf{x} + b)).$$

You should think of $\mathbf{w}$ and b as representing a hyperplane which separates the positive data from the negative data. This hyperplane is known as the **decision boundary**. The particular rule is given by the choice of $\mathbf{w}$ and b.

14.2.1 Why a linear rule?

This family of rules may look bad to you. It is easy to come up with examples that it misclassifies badly. The rule has important strengths: it is easy to estimate the best choice of rule, it works very well in practice on real data, and it is fast to evaluate. For practical examples, experience shows that the error rate can be improved by adding features to the vector $\mathbf{x}$.

Example: 14.1 *A linear model with a single feature*

Assume we use a linear model with one feature. Then the model has the form $y_i^{(p)} = \text{sign}(ax_i + b)$. For any particular example which has the feature value x^*, this means we will test whether x^* is larger than, or smaller than, $-b/a$.

Example: 14.2 *A linear model with two features*

Assume we use a linear model with two features. Then the model has the form $y_i^{(p)} = \text{sign}(\mathbf{a}^T\mathbf{x}_i + b)$. The sign changes along the line $\mathbf{a}^T\mathbf{x} + b = 0$. You should check that this is, indeed, a line. On one side of this line, the model makes positive predictions; on the other, negative. Which side is which can be swapped by multiplying $\mathbf{a}$ and b by -1.

14.2.2 Logistic Regression

We will choose $\mathbf{a}$ and b by choosing values that minimize the cost of errors made by the classifier. In particular, we will adopt a cost function of the form:

$$\text{Training error cost} + \text{penalty term}.$$

For the moment, we will ignore the penalty term. The training error cost will be of the form

$$(1/N)\sum_{i=1}^{N} C((\mathbf{a}^T\mathbf{x}_i + b), y_i)$$

so at each point in the training data, we compute a cost from the true value of y_i and the predicted value. This cost should be large if y_i and $y_i^{(p)}(\mathbf{a}^T\mathbf{x}_i + b, y_i)$ have different signs, and small if they have the same sign. It is convenient to write

$$\gamma_i = (\mathbf{a}^T\mathbf{x}_i + b, y_i).$$

For **logistic regression**, the cost function using this notation is

$$C((\mathbf{a}^T\mathbf{x}_i + b), y_i) = C(\gamma_i, y_i) = \log\left(1 + \exp\left(-y_i\gamma_i\right)\right).$$

The function $L(1, \gamma)$ is plotted in Figure 14.1. This loss is sometimes known as the **logistic loss**. This loss very strongly penalizes a large positive γ_i if y_i is negative (and vice versa). However, there is no significant advantage to having a large positive γ_i if y_i is positive. This means that the significant components of the loss function will be due to examples that the classifier gets wrong, but also due to examples that have γ_i near zero (i.e., the example is close to the decision boundary).

You should notice another important property of this loss. Assume we wish to predict a label for a new data item. The loss we would incur depends quite strongly on the magnitude of γ. If we produce a large value of γ for that data item *with the wrong sign*, then we would incur a very large loss. This means that we should prefer values of $\mathbf{a}$ and b that will tend to produce small values of γ. In turn, we should prefer small values of $\mathbf{a}$ if they give about the same value of training loss. This is our penalty term. We should use a cost function of the form

$$\text{Training Loss} + \frac{\lambda}{2}\left(\text{Norm of } \mathbf{a}\right)$$

which is

$$\left[\frac{1}{N}\sum_{i \in \text{examples}} \{\log(1 + \exp -y_i\gamma_i\})\right] + \frac{\lambda}{2}\mathbf{a}^T\mathbf{a}$$

where $\lambda > 0$ is a constant chosen for good performance. Too large a value of λ, and the classifier will behave poorly on training and test data; too small a value, and the classifier will behave poorly on test data.

Usually, the value of λ is set with a validation dataset. We train classifiers with different values of λ on a test dataset, then evaluate them on a validation set—data whose labels are known, but which is not used for training—and finally choose the λ that gets the best validation error. The error is not usually all that sensitive to the choice of λ, so searching values by factors of 10 is usually fine.

The penalty term is often referred to as a **regularizer**, because it tends to discourage solutions that are large (and so have possible high loss on future test data) but are not strongly supported by the training data.

14.2.3 The Hinge Loss

There are alternate losses, that are very useful. The linear **support vector machine** or **SVM** uses the **hinge loss**. In this case the loss comparing the label value y_i and the prediction $\gamma_i = (\mathbf{w} \cdot \mathbf{x}_i + b)$ can be written as

$$L_h(y_i, \gamma_i) = \max(0, 1 - y_i\gamma_i).$$

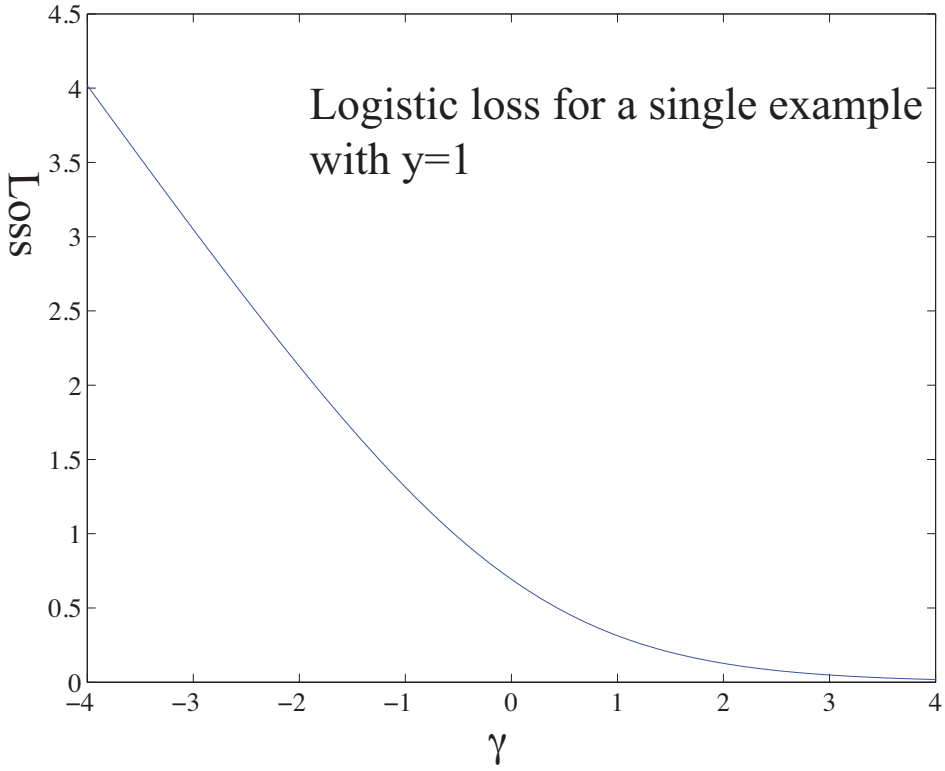

FIGURE 14.1: *The logistic loss, plotted for the case $y_i = 1$. In the case of the logistic loss, the horizontal variable is the $\gamma_i = \mathbf{a} \cdot \mathbf{x}_i$ of the text. Notice that giving a strong negative response to this positive example causes a loss that grows linearly as the magnitude of the response grows. Notice also that giving an insufficiently positive response also causes a loss. Giving a strongly positive response is cheap or free.*

This loss is always non-negative. For the moment, assume $y_i = 1$; then, any prediction by the classifier with value greater than one will incur no loss, and any smaller prediction will incur a cost that is linear in the prediction value (Figure **??**). This means that minimizing the loss will encourage the classifier to (a) make strong positive (or negative) predictions for positive (or negative) examples and (b) for examples it gets wrong, make the most positive (negative) prediction that it can. The expression

$$\left[(1/N)\sum_{i=1}^{N}\max(0, 1 - y_i(\mathbf{w}\cdot\mathbf{x}_i + b))\right] + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}$$

fits into the rule of above, where we obtained a classifier by minimizing

$$\text{Training Loss} + \text{Regularizer}.$$

We have just changed the loss.

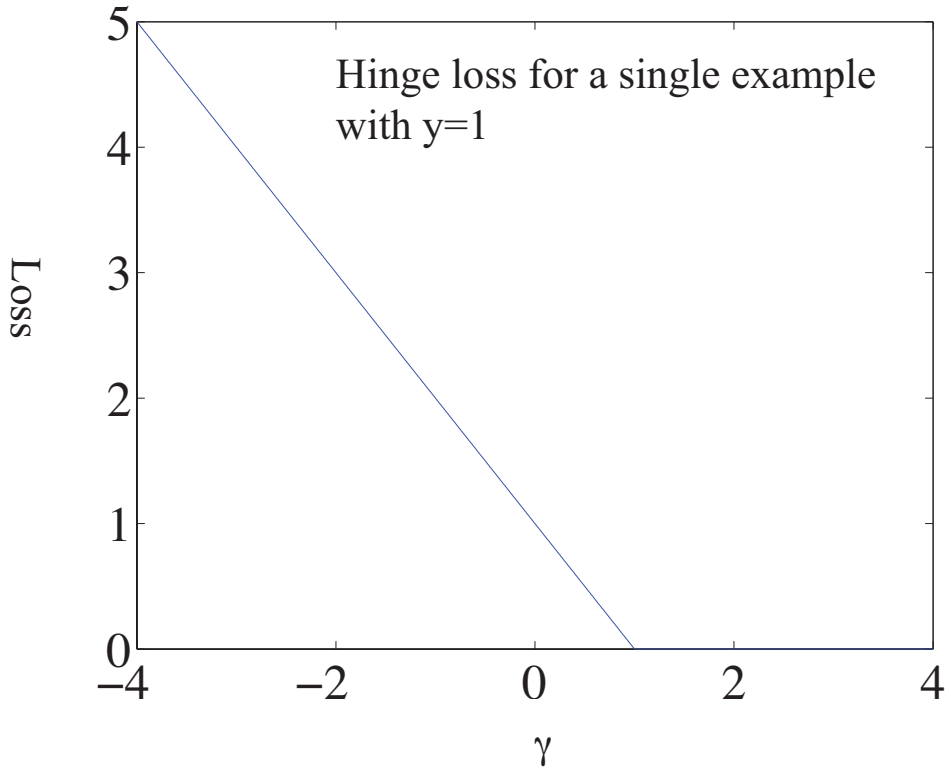

FIGURE 14.2: *The hinge loss, plotted for the case $y_i = 1$. The horizontal variable is the $\gamma_i = \mathbf{a} \cdot \mathbf{x}_i$ of the text. Notice that giving a strong negative response to this positive example causes a loss that grows linearly as the magnitude of the response grows. Notice also that giving an insufficiently positive response also causes a loss. Giving a strongly positive response is or free. The loss should look a lot like the hinge loss to you.*

14.3 BASIC IDEAS FOR NUMERICAL MINIMIZATION

We must now obtain a classifier that minimizes either logistic or hinge loss. Assume we have a function $g(\mathbf{a})$, and we wish to obtain a value of $\mathbf{a}$ that achieves the minimum for that function. Sometimes we can solve this problem in closed form by constructing the gradient and finding a value of $\mathbf{a}$ the makes the gradient zero. More usually we need a numerical method. Implementing these numerical methods is a specialized business, and it is usual to use general optimization codes. This section is intended to sketch how such codes work, so you can read manual pages, etc. more effectively. Personally, I am a happy user of Matlab's `fminunc`, although the many different settings take some getting used to.

14.3.1 Overview

Typical codes take a description of the objective function (typically, the name of a function), a start point for the search, and a collection of parameters. All codes take an estimate $\mathbf{a}^{(i)}$, update it to $\mathbf{a}^{(i+1)}$, then check to see whether the result is a minimum. This process is started from the start point. The update is usually obtained by computing a direction $\mathbf{p}^{(i)}$ such that for small values of h, $g(\mathbf{a}^{(i)}+h\mathbf{p}^{(i)})$ is smaller than $g(\mathbf{a}^{(i)})$. Such a direction is known as a **descent direction**.

Assume we have a descent direction. We must now choose how far to travel along that direction. We can see $g(\mathbf{a}^{(i)}+h\mathbf{p}^{(i)})$ as a function of h. Write this function as $\phi(h)$. We start at $h = 0$ (which is the original value $\mathbf{a}^{(i)}$, so $\phi(0) = g(\mathbf{a}^{(i)})$, and move in the direction of increasing h to find a small value of $\phi(h)$ that is less than $\phi(0)$. The descent direction was chosen so that for small $h > 0$, $\phi(h) < \phi(0)$; one way to tell we are at a minimum is we cannot choose a descent direction. Searching for a good value of h is known as **line search**. Typically, this search involves a sequence of estimated values of h, which we write h_i. One algorithm is to start with (say) $h_0 = 1$; if $\phi(h_i)$ is not small enough (and there are other tests we may need to apply — this is a summary!), we compute $h_{(i+1)} = (1/2)h_i$. This stops when some h_i passes a test, or when it is so small that the step is pointless.

14.3.2 Gradient Descent

One method to choose a descent direction is **gradient descent**, which uses the negative gradient of the function. Recall our notation that

$$\mathbf{a} = \begin{pmatrix} a_1 \\ a_2 \\ \dots \\ a_d \end{pmatrix}$$

and that

$$\nabla g = \begin{pmatrix} \frac{\partial g}{\partial a_1} \\ \frac{\partial g}{\partial a_2} \\ \dots \\ \frac{\partial g}{\partial a_d} \end{pmatrix}.$$

We can write a Taylor series expansion for the function $g(\mathbf{a}^{(i)} + h\mathbf{p}^{(i)})$. We have that

$$g(\mathbf{a}^{(i)} + h\mathbf{p}^{(i)}) = g(\mathbf{a}^{(i)}) + h(\nabla g)^T\mathbf{p}^{(i)} + O(h^2)$$

This means that we can expect that if

$$\mathbf{p}^{(i)} = -\nabla g(\mathbf{a}^{(i)}),$$

we expect that, at least for small values of h, $g(\mathbf{a}^{(i)}+h\mathbf{p}^{(i)})$ will be less than $g(\mathbf{a}^{(i)})$.

This works (as long as g is differentiable, and quite often when it isn't) because g must go down for at least small steps in this direction. There are two ways to evaluate a gradient. You can require that the software estimate a numerical derivative for you, which usually slows things down somewhat, or you can supply a

gradient value. Usually this gradient value must be computed by the same function that computes the objective function value.

One tip: in my experience, about 99% of problems with numerical optimization codes occur because the user didn't check that the gradient their function computed is right. Most codes will compute a numerical gradient for you, then check that against your gradient; if they're sufficiently different, the code will complain. You don't want to do this at runtime, because it slows things up, but it's an excellent idea to check.

14.3.3 Stochastic Gradient Descent

Assume we wish to minimize some function $g(\mathbf{a}) = g_0(\mathbf{a}) + (1/N)\sum_{i=1}^{N} g_i(\mathbf{a})$, as a function of $\mathbf{a}$. Gradient descent would require us to form

$$-\nabla g(\mathbf{a}) = -\left(\nabla g_0(\mathbf{a}) + (1/N)\sum_{i=1}^{N} \nabla g_i(\mathbf{a})\right)$$

and then take a small step in this direction. But if N is large, this is unattractive, as we might have to sum a lot of terms. This happens a lot in building classifiers, where you might quite reasonably expect to deal with millions of examples. Touching each example at each step really is impractical.

Instead, assume that, at each step, we choose a number k in the range $1 \ldots N$ uniformly and at random, and form

$$\mathbf{p}_k = -\left(\nabla g_0(\mathbf{a}) + \nabla g_k(\mathbf{a})\right)$$

and then take a small step along $\mathbf{p}_k$. Our new point becomes

$$\mathbf{a}^{(i+1)} = \mathbf{a}^{(i)} + \eta \mathbf{p}_k^{(i)},$$

where η is called the **steplength** (even though it very often isn't the length of the step we take!). It is easy to show that

$$\mathbb{E}[\mathbf{p}_k] = \nabla g(\mathbf{a})$$

(where the expectation is over the random choice of k). This implies that if we take many small steps along $\mathbf{p}_k$, they should average out to a step backwards along the gradient. This approach is known as **stochastic gradient descent** (because we're not going along the gradient, but along a random vector which is the gradient only in expectation). It isn't obvious that stochastic gradient descent is a good idea. Although each step is easy to take, we may need to take more steps. The question is then whether we gain in the increased speed of the step what we lose by having to take more steps. Not much is known theoretically, but in practice the approach is hugely successful for training classifiers.

Choosing a steplength η takes some work. Line search won't work, because we don't want to evaluate the function g, because doing so involves looking at each of the g_i terms. Instead, one uses a steplength that is large at the start — so that it can explore large changes in the values of the classifier parameters — and small steps later — so that it settles down. One useful strategy is to divide training into

epochs. Each epoch is a block of a fixed number of iterations. Each iteration is one of the steps given above, with fixed steplength. However, the steplength changes from epoch to epoch. In particular, in the r'th epoch, the steplength is

$$\eta^{(r)} = \frac{a}{r+b}$$

where a and b are constants chosen by experiment with small subsets of the dataset.

One cannot really test whether stochastic gradient descent has converged to the right answer. A better approach is to plot the error as a function of epoch on a validation set. This should vary randomly, but generally go down as the epochs proceed.

14.3.4 Example: Training a Support Vector Machine with Stochastic Gradient Descent

We need to choose $\mathbf{w}$ and b to minimize

$$C(\mathbf{w}, b) = (1/N) \sum_{i=1}^{N} \max(0, 1 - y_i (\mathbf{w} \cdot \mathbf{x}_i + b)) + \frac{\lambda}{2} \mathbf{w}^T \mathbf{w}.$$

There are several methods to do so. Section 1 describes some of the many available support vector machine training packages on the web; it is often, even usually, a good idea to use one of these. But it is worth understanding how such things work.

For a support vector machine, stochastic gradient descent is particularly easy. We have estimates $\mathbf{w}^{(n)}$ and $b^{(n)}$ of the classifier parameters, and we want to improve the estimates. We pick the k'th example at random. We must now compute

$$\nabla \left(\max(0, 1 - y_k (\mathbf{w} \cdot \mathbf{x}_k + b)) + \frac{\lambda}{2} \mathbf{w}^T \mathbf{w} \right).$$

Assume that $y_k (\mathbf{w} \cdot \mathbf{x}_k + b) > 1$. In this case, the classifier predicts a score with the right sign, and a magnitude that is greater than one. Then the first term is zero, and the gradient of the second term is easy. Now if $y_k (\mathbf{w} \cdot \mathbf{x}_k + b) < 1$, we can ignore the max, and the first term is $1 - y_k (\mathbf{w} \cdot \mathbf{x}_k + b)$; the gradient is again easy. But what if $y_k (\mathbf{w} \cdot \mathbf{x}_k + b) = 1$? there are two distinct values we could choose for the gradient, because the max term isn't differentiable. It turns out not to matter which term we choose (Figure **??**), so we can write the gradient as

$$p_k = \begin{cases} \begin{bmatrix} \lambda \mathbf{w} \\ 0 \end{bmatrix} & \text{if } y_k (\mathbf{w} \cdot \mathbf{x}_k + b) \geq 1 \\ \begin{bmatrix} \lambda \mathbf{w} - y_k \mathbf{x} \\ -y_k \end{bmatrix} & \text{otherwise} \end{cases}$$

We choose a steplength η, and update our estimates using this gradient. This yields:

$$\mathbf{w}^{(n+1)} = \mathbf{w}^{(n)} - \eta \begin{cases} \lambda \mathbf{w} & \text{if } y_k (\mathbf{w} \cdot \mathbf{x}_k + b) \geq 1 \\ \lambda \mathbf{w} - y_k \mathbf{x} & \text{otherwise} \end{cases}$$

and

$$b^{(n+1)} = b^{(n)} - \eta \begin{cases} 0 & \text{if } y_k (\mathbf{w} \cdot \mathbf{x}_k + b) \geq 1 \\ -y_k & \text{otherwise} \end{cases}.$$

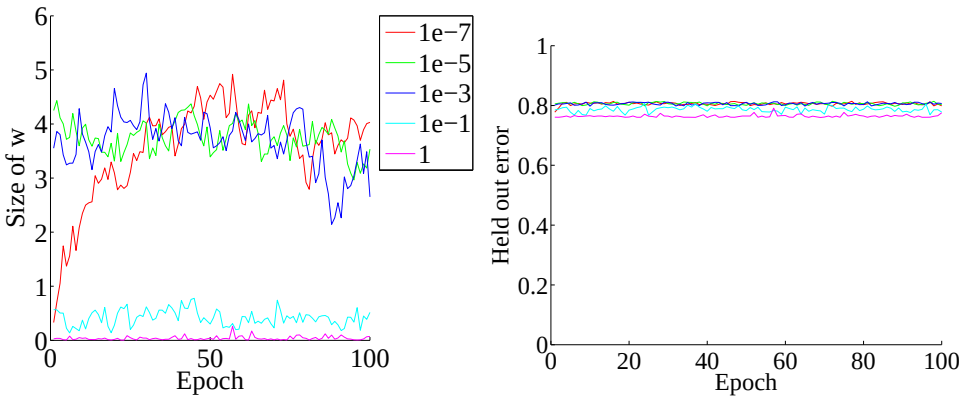

FIGURE 14.3: *On the* **left**, *the magnitude of the weight vector* $\mathbf{w}$ *at the end of each epoch for the first training regime described in the text. On the* **right**, *the accuracy on held out data at the end of each epoch. Notice how different choices of regularization parameter lead to different magnitudes of* $\mathbf{w}$*; how the method isn't particularly sensitive to choice of regularization parameter (they change by factors of 100); how the accuracy settles down fairly quickly; and how overlarge values of the regularization parameter do lead to a loss of accuracy.*

To construct figures, I downloaded the dataset at `http://archive.ics.uci.edu/ml/datasets/Adult`. This dataset apparently contains 48, 842 data items, but I worked with only the first 32, 000. Each consists of a set of numeric and categorical features describing a person, together with whether their annual income is larger than or smaller than 50K\$. I ignored the categorical features to prepare these figures. This isn't wise if you want a good classifier, but it's fine for an example. I used these features to predict whether income is over or under 50K\$. I split the data into 5, 000 test examples, and 27,000 training examples. It's important to do so at random. There are 6 numerical features. I subtracted the mean (which doesn't usually make much difference) and rescaled each so that the variance was 1 (which is often very important). I used two different training regimes.

In the first training regime, there were 100 epochs. In each epoch, I applied 426 steps. For each step, I selected one data item uniformly at random (sampling with replacement), then stepped down the gradient. This means the method sees a total of 42, 600 data items. This means that there is a high probability it has touched each data item once (27, 000 isn't enough, because we are sampling with replacement, so some items get seen more than once). I chose 5 different values for the regularization parameter and trained with a steplength of $1/(0.01 * e + 50)$, where e is the epoch. At the end of each epoch, I computed $\mathbf{w}^T\mathbf{w}$ and the accuracy (fraction of examples correctly classified) of the current classifier on the held out test examples. Figure 14.3 shows the results. You should notice that the accuracy changes slightly each epoch; that for larger regularizer values $\mathbf{w}^T\mathbf{w}$ is smaller; and that the accuracy settles down to about 0.8 very quickly.

In the second training regime, there were 100 epochs. In each epoch, I applied

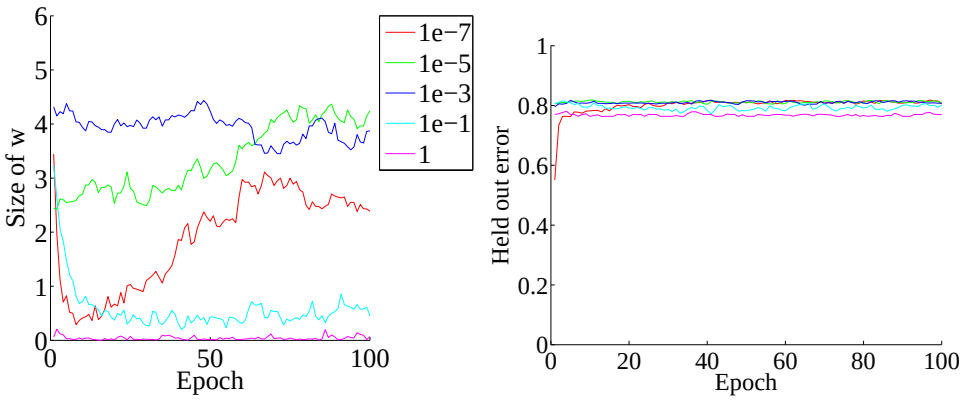

FIGURE 14.4: *On the* **left**, *the magnitude of the weight vector* $\mathbf{w}$ *at the end of each epoch for the second training regime described in the text. On the* **right**, *the accuracy on held out data at the end of each epoch. Notice how different choices of regularization parameter lead to different magnitudes of* $\mathbf{w}$*; how the method isn't particularly sensitive to choice of regularization parameter (they change by factors of 100); how the accuracy settles down fairly quickly; and how overlarge values of the regularization parameter do lead to a loss of accuracy.*

50 steps. For each step, I selected one data item uniformly at random (sampling with replacement), then stepped down the gradient. This means the method sees a total of 5,000 data items, and about 3, 216 unique data items — it hasn't seen the whole training set. I chose 5 different values for the regularization parameter and trained with a steplength of $1/(0.01 * e + 50)$, where e is the epoch. At the end of each epoch, I computed $\mathbf{w}^T\mathbf{w}$ and the accuracy (fraction of examples correctly classified) of the current classifier on the held out test examples. Figure 14.4 shows the results. You should notice that the accuracy changes slightly each epoch; that for larger regularizer values $\mathbf{w}^T\mathbf{w}$ is smaller; and that the accuracy settles down to about 0.8 very quickly; and that there isn't much difference between the two training regimes. All of these points are relatively typical of stochastic gradient descent with very large datasets.

14.4 CLASSIFYING WITH RANDOM FORESTS

I described a classifier as a rule that takes a feature, and produces a class. One way to build such a rule is with a sequence of simple tests, where each test is allowed to use the results of all previous tests. This class of rule can be drawn as a tree (Figure **??**), where each node represents a test, and the edges represent the possible outcomes of the test. To classify a test item with such a tree, you present it to the first node; the outcome of the test determines which node it goes to next; and so on, until the example arrives at a leaf. When it does arrive at a leaf, we label the test item with the most common label in the leaf. This object is known as a **decision tree**.

Figure **??** shows a simple 2D dataset with four classes, next to a decision tree

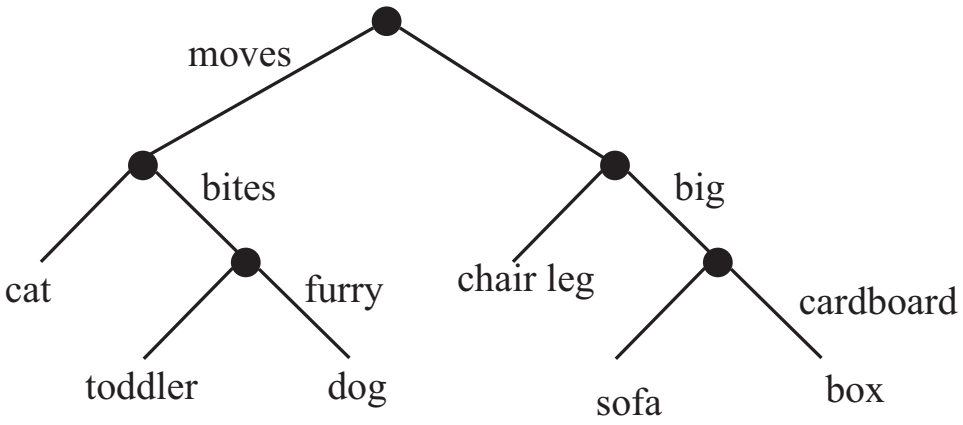

FIGURE 14.5: *This — the household robot's guide to obstacles — is a typical decision tree. I have labelled only one of the outgoing branches, because the other is the negation. So if the obstacle moves, bites, but isn't furry, then it's a toddler. In general, an item is passed down the tree until it hits a leaf. It is then labelled with the leaf's label.*

that will correctly classify at least the training data. Actually classifying data with a tree like this is straightforward. We take the data item, and pass it down the tree. Notice it can't go both left and right, because of the way the tests work. This means each data item arrives at a single leaf. We take the most common label at the leaf, and give that to the test item.

The important question is how to get the tree from data. It turns out that the best approach for building a tree incorporates a great deal of randomness. As a result, we will get a different tree each time we train a tree on a dataset. None of the individual trees will be particularly good (they are often referred to as "weak learners"). The natural thing to do is to produce many such trees (a **decision forest**), and allow each to vote; the class that gets the most votes, wins. This strategy is extremely effective.

14.4.1 Building a Decision Tree

There are many algorithms for building decision trees. We will use an approach chosen for simplicity and effectiveness; be aware there are others. We will always use a binary tree, because it's easier to describe and because that's usual (it doesn't change anything important, though). Each node has a **decision function**, which takes data items and returns either 1 or -1.

We train the tree by thinking about its effect on the training data. We pass the whole pool of training data into the root. Any node splits its incoming data into two pools, left (all the data that the decision function labels 1) and right (ditto, -1). Finally, each leaf contains a pool of data, which it can't split because it is a leaf.

Training the tree uses a straightforward algorithm. First, we choose a class of

decision functions to use at each node. It turns out that a very effective algorithm is to choose a single feature at random, then test whether its value is larger than, or smaller than a threshold. For this approach to work, one needs to be quite careful about the choice of threshold, which is what we describe in the next section. Some minor adjustments, described below, are required if the feature chosen isn't ordinal. Surprisingly, being clever about the choice of *feature* doesn't seem add a great deal of value. We won't spend more time on other kinds of decision function, though there are lots.

Now assume we use a decision function as described, and we know how to choose a threshold. We start with the root node, then recursively either split the pool of data at that node, passing the left pool left and the right pool right, or stop splitting and return. Splitting involves choosing a decision function from the class to give the "best" split for a leaf. The main questions are how to choose the best split (next section), and when to stop.

Stopping is relatively straightforward. Quite simple strategies for stopping are very good. It is hard to choose a decision function with very little data, so we must stop splitting when there is too little data at a node. We can tell this is the case by testing the amount of data against a threshold, chosen by experiment. If all the data at a node belongs to a single class, there is no point in splitting. Finally, constructing a tree that is too deep tends to result in generalization problems, so we usually allow no more than a fixed depth D of splits. Choosing the best splitting threshold is more complicated.

14.4.2 Entropy and Information Gain

Figure 14.6 shows two possible splits of a pool of training data. These splits are obtained by testing the horizontal feature against a threshold. In one case, the left and the right pools contain about the same fraction of positive ('x') and negative ('o') examples. In the other, the left pool is all positive, and the right pool is mostly negative. Clearly this is the better choice of threshold. But we need some way to score what has happened, so we can tell which threshold is best. Notice that, in the uninformative case, knowing that a data item is on the left (or the right) does not tell me much more about the data than I already knew. This is because

$$p(1|\text{left pool}) \approx p(1|\text{parent pool}).$$

In the second case, knowing a data item is on the left classifies it completely. In this case, my uncertainty about what class the data item belongs to is significantly reduced if I know whether it goes left or right. To choose a good threshold, we need to keep track of how informative the split is.

It turns out to be straightforward to keep track of information, in simple cases. We will start with an example. Assume I have 4 classes. There are 8 examples in class 1, 4 in class 2, 2 in class 3, and 2 in class 4. How much information *on average* will you need to send me to tell me the class of a given example? Clearly, this depends on how you communicate the information. You could send me the complete works of Edward Gibbon to communicate class 1; the Encyclopaedia for class 2; and so on. But this would be redundant. The question is how little can you send me. Keeping track of the amount of information is easier if we encode it

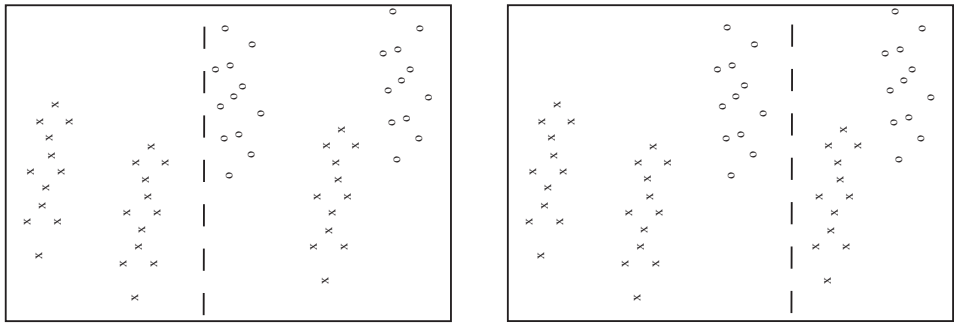

FIGURE 14.6: *Two possible splits of a pool of training data. Positive data is represented with an 'x', negative data with a 'o'. Notice that if we split this pool with the informative line, all the points on the left are 'x's, and two-thirds of the points on the right are 'o's. This means that knowing which side of the split a point lies would give us a good basis for estimating the label. In the less informative case, about two-thirds of the points on the left are 'x's and about half on the right are 'x's — knowing which side of the split a point lies is much less useful in deciding what the label is.*

with bits (i.e. you can send me sequences of '0's and '1's).

Imagine the following scheme. If an example is in class 1, you send me a '1'. If it is in class 2, you send me '01'; if it is in class 3, you send me '001'; and in class 4, you send me '101'. Then the expected number of bits you will send me is

$$p(\text{class} = 1)1 + p(2)2 + p(3)3 + p(4)3 = \frac{1}{2}1 + \frac{1}{4}2 + \frac{1}{8}3 + \frac{1}{8}3$$

which is 1.75 bits. This number doesn't have to be an integer, because it's an expectation.

Notice that for the i'th class, you have sent me $-\log_2 p(i)$ bits. We can write the expected number of bits you need to send me as

$$-\sum_i p(i) \log_2 p(i).$$

This expression handles other simple cases correctly, too. You should try what happens if you have two classes, each with 8 examples in them; 256 classes, each with one example in them; and 5 classes, with 16 examples in class 1, 8 in class 2, etc. If you try other examples, you may find it hard to construct a scheme where you can send as few bits *on average* as this expression predicts. It turns out that, in general, the smallest number of bits you will need to send me is given by the expression

$$-\sum_i p(i) \log_2 p(i)$$

under all conditions, though it may be hard or impossible to determine what representation is required to achieve this number.

Now we return to the splits. Write $\mathcal{P}$ for the set of all data at the node. Write $\mathcal{P}_l$ for the left pool, and $\mathcal{P}_r$ for the right pool. The **entropy** of a pool $\mathcal{C}$ is a function $H(\mathcal{C})$ that scores how many bits would be required to represent the class of an item in that pool, on average. Write $n(i; \mathcal{C})$ for the number of items of class i in the pool, and $N(\mathcal{C})$ for the number of items in the pool. Then the entropy of the pool $\mathcal{C}$ is

$$-\sum_i \frac{n(i;\mathcal{C})}{N(\mathcal{C})} \log_2 \frac{n(i;\mathcal{C})}{N(\mathcal{C}}.$$

It is straightforward that $H(\mathcal{P})$ bits are required to classify an item in the parent pool $\mathcal{P}$. For an item in the left pool, we need $H(\mathcal{P}_l)$ bits; for an item in the right pool, we need $H(\mathcal{P}_r)$ bits. If we split the parent pool, we expect to encounter items in the left pool with probability

$$\frac{N(\mathcal{P}_l)}{N(\mathcal{P})}$$

and items in the right pool with probability

$$\frac{N(\mathcal{P}_r)}{N(\mathcal{P})}.$$

This means that, on average, we must supply

$$\frac{N(\mathcal{P}_l)}{N(\mathcal{P})} H(\mathcal{P}_l) + \frac{N(\mathcal{P}_r)}{N(\mathcal{P})} H(\mathcal{P}_r)$$

bits to classify data items if we split the parent pool. Now a good split is one that results in left and right pools that are informative. In turn, we should need fewer bits to classify once we have split than we need before the split. You can see the difference

$$I(\mathcal{P}_l, \mathcal{P}_r; \mathcal{P}) = H(\mathcal{P}) - \left(\frac{N(\mathcal{P}_l)}{N(\mathcal{P})} H(\mathcal{P}_l) + \frac{N(\mathcal{P}_r)}{N(\mathcal{P})} H(\mathcal{P}_r) \right)$$

as the **information gain** caused by the split. This is the average number of bits that you *don't* have to supply if you know which side of the split an example lies. Better splits have larger information gain.

14.4.3 Choosing a Split with Information Gain

Recall that our decision function is to choose a feature at random, then test its value against a threshold. Any data point where the value is larger goes to the left pool; where the value is smaller goes to the right. This may sound much too simple to work, but it is actually effective and popular. Assume that we are at a node, which we will label k. We have the pool of training examples that have reached that node. The i'th example has a feature vector $\mathbf{x}_i$, and each of these feature vectors is a d dimensional vector.

We choose an integer j in the range $1 \ldots d$ uniformly and at random. We will split on this feature, and we store j in the node. Recall we write $x_i^{(j)}$ for the value of the j'th component of the i'th feature vector. We will choose a threshold t_k,

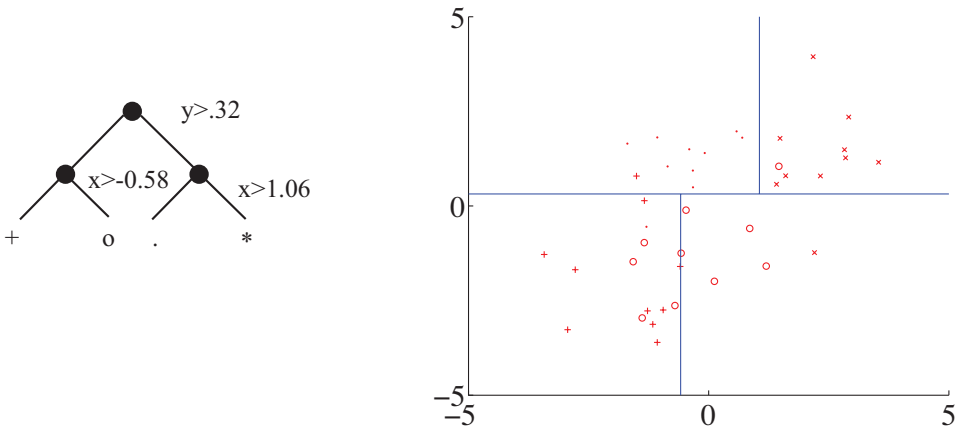

FIGURE 14.7: *A straightforward decision tree*

and split by testing the sign of $x_i^{(j)} - t_k$. Choosing the value of t_k is easy. Assume there are N_k examples in the pool. Then there are $N_k - 1$ possible values of t_k that lead to different splits. To see this, sort the N_k examples by $x^{(j)}$, then choose values of t_k halfway between example values (Figure **??**). For each of these values, we compute the information gain of the split. We then keep the threshold with the best information gain.

We can elaborate this procedure in a useful way, by choosing m features at random, finding the best split for each, then keeping the feature and threshold value that is best. It is important that m is a lot smaller than the total number of features — a usual root of thumb is that m is about the square root of the total number of features. It is usual to choose a single m, and choose that for all the splits.

Now assume we happen to have chosen to work with a feature that isn't ordinal, and so can't be tested against a threshold. A natural, and effective, strategy is as follows. We can split such a feature into two pools by flipping an unbiased coin for each value — if the coin comes up H, any data point with that value goes left, and if it comes up T, any data point with that value goes right. We chose this split at random, so it might not be any good. We can come up with a good split by repeating this procedure F times, computing the information gain for each split, then keeping the one that has the best information gain. We choose F in advance, and it usually depends on the number of values the categorical variable can take.

We now have a relatively straightforward blueprint for an algorithm, which I have put in a box. It's a blueprint, because there are a variety of ways in which it can be revised and changed.

Procedure: 14.1 *Building a decision tree*

Assume we have a data set

TODO: an algorithm block

14.4.4 Forests

A single decision tree tends to yield poor classifications. One reason is because the tree is not chosen to give the best classification of its training data. We used a random selection of splitting variables at each node, so the tree can't be the "best possible". Obtaining the best possible tree presents significant technical difficulties. It turns out that the tree that gives the best possible results on the training data can perform rather poorly on test data. The training data is a small subset of possible examples, and so must differ from the test data. The best possible tree on the training data might have a large number of small leaves, built using carefully chosen splits. But the choices that are best for training data might not be best for test data.

Rather than build the best possible tree, we have built a tree efficiently, but with number of random choices. If we were to rebuild the tree, we would obtain a different result. This suggests the following extremely effective strategy: build many trees, and classify by merging their results.

14.4.5 Building and Evaluating a Decision Forest

There are two important strategies for building and evaluating decision forests. I am not aware of evidence strongly favoring one over the other, but different software packages use different strategies, and you should be aware of the options. In one strategy, we separate labelled data into a training and a test set. We then build multiple decision trees, training each using the whole training set. Finally, we evaluate the forest on the test set. In this approach, the forest has not seen some fraction of the available labelled data, because we used it to test. However, each tree has seen every training data item.

In the other strategy, sometimes called **bagging**, each time we train a tree we randomly subsample the labelled data with replacement, to yield a training set the same size as the original set of labelled data. Notice that there will be duplicates in this training set, which is like a bootstrap replicate. This training set is often called a **bag**. We keep a record of the examples that do not appear in the bag (the "out of bag" examples). Now to evaluate the forest, we evaluate each tree on its out of bag examples, and average these error terms. In this approach, the entire forest has seen all labelled data, and we also get an estimate of error, but no tree has seen all the training data.

14.4.6 Classifying Data Items with a Decision Forest

Once we have a forest, we must classify test data items. There are two major strategies. The simplest is to classify the item with each tree in the forest, then take the class with the most votes. This is effective, but discounts some evidence that might be important. For example, imagine one of the trees in the forest has a leaf with many data items with the same class label; another tree has a leaf with exactly one data item in it. One might not want each leaf to have the same vote.

An alternative strategy that takes this observation into account is to pass the test data item down each tree. When it arrives at a leaf, we record one vote for each of the training data items in that leaf. The vote goes to the class of the training data item. Finally, we take the class with the most votes. This approach allows big, accurate leaves to dominate the voting process. Both strategies are in use, and I am not aware of compelling evidence that one is always better than the other. This may be because the randomness in the training process makes big, accurate leaves uncommon in practice.

Notice one of the major attractions of random forests. Our strategy doesn't depend on the number of classes we are dealing with (though the results might).

Worked example 14.1 *Classifying heart disease data*

Build a random forest classifier to classify the "heart" dataset from the UC Irvine machine learning repository. The dataset is at `http://archive.ics.uci.edu/ml/datasets/Heart+Disease`. There are several versions. You should look at the processed Cleveland data, which is in the file "processed.cleveland.data.txt".

Solution: I used the R random forest package. This uses a bagging strategy. There is sample code in listing **??**. This package makes it quite simple to fit a random forest, as you can see. In this dataset, variable 14 (V14) takes the value 0, 1, 2, 3 or 4 depending on the severity of the narrowing of the arteries. Other variables are physiological and physical measurements pertaining to the patient (read the details on the website). I tried to predict all five levels of variable 14, using the random forest as a multivariate classifier. This works rather poorly, as the out-of-bag class confusion matrix below shows. The total out-of-bag error rate was 45%.

	Predict 0	Predict 1	Predict 2	Predict 3	Predict 4	Class error
True 0	151	7	2	3	1	7.9%
True 1	32	5	9	9	0	91%
True 2	10	9	7	9	1	81%
True 3	6	13	9	5	2	86%
True 4	2	3	2	6	0	100%

This is the example of a class confusion matrix from table 14.1. Fairly clearly, one can predict narrowing or no narrowing from the features, but not the degree of narrowing (at least, not with a random forest). So it is natural to quantize variable 14 to two levels, 0 (meaning no narrowing), and 1 (meaning any narrowing, so the original value could have been 1, 2, or 3). I then built a random forest to predict this from the other variables. The total out-of-bag error rate was 19%, and I obtained the following out-of-bag class confusion matrix

	Predict 0	Predict 1	Class error
True 0	138	26	16%
True 1	31	108	22%

Notice that the false positive rate (16%, from 26/164) is rather better than the false negative rate (22%). Looking at these class confusion matrices, you might wonder whether it is better to predict $0, \ldots, 4$, then quantize. But this is not a particularly good idea. While the false positive rate is 7.9%, the false negative rate is much higher (36%, from 50/139). In this application, a false negative is likely more of a problem than a false positive, so the tradeoff is unattractive.

Listing 14.1: R code used for the random forests of worked example 1

```
setwd('/users/daf/Current/courses/Probcourse/Trees/RCode');
install.packages('randomForest')
library(randomForest)
heart<-read.csv('processed.cleveland.data.txt', header=FALSE)
heart$levels<-as.factor(heart$V14)
heartforest.allvals<-
  randomForest(formula=levels~V1+V2+V3+V4+V5+V6
               +V7+V8+V9+V10+V11+V12+V13,
               data=heart, type='classification', mtry=5)
# this fits to all levels
# I got the CCM by typing
heartforest.allvals
heart$yesno<-cut(heart$V14, c(-Inf, 0.1, Inf))
heartforest<-
  randomForest(formula=yesno~V1+V2+V3+V4+V5+V6
               +V7+V8+V9+V10+V11+V12+V13,
               data=heart, type='classification', mtry=5)
# this fits to the quantized case
# I got the CCM by typing
heartforest
```

14.5 PRACTICAL METHODS FOR BUILDING CLASSIFIERS

We have described several apparently very different classifiers here. But which classifier should one use for a particular application? Generally, this should be dealt with as a practical rather than a conceptual question: that is, one tries several, and uses the one that works best. With all that said, experience suggests that the first thing to try for most problems is a linear SVM or logistic regression, which tends to be much the same thing. Nearest neighbor strategies are always useful, and are consistently competitive with other approaches when there is lots of training data and one has some idea of appropriate relative scaling of the features. The main difficulty with nearest neighbors is actually finding the nearest neighbors of a query. Approximate methods are now very good, and are reviewed in Section 15.4.1. The attraction of these methods is that it is relatively easy to build multi-class classifiers, and to add new classes to a system of classifiers.

14.5.1 Manipulating Training Data to Improve Performance

Generally, more training data leads to a better classifier. However, training classifiers with large datasets can be difficult, and it can be hard to get enough training data. Typically, only a relatively small number of example items are really important in determining the behavior of a classifier (we see this phenomenon in greater detail in Section **??**). The really important examples tend to be rare cases that are quite hard to discriminate. This is because these cases affect the position of the decision boundary most significantly. We need a large dataset to ensure that these cases are present.

There are some useful tricks that help.

We train on a subset of the examples, run the resulting classifier on the rest of the examples, and then insert the false positives and false negatives into the training

set to retrain the classifier. This is because the false positives and false negatives are the cases that give the most information about errors in the configuration of the decision boundaries. We may repeat this several times, and in the final stages, we may use the classifier to seek false positives. For example, we might collect pictures from the Web, classify them, and then look at the positives for errors. This strategy is sometimes called **bootstrapping** (the name is potentially confusing because there is an unrelated statistical procedure known as bootstrapping; nonetheless, we're stuck with it at this point).

There is an extremely important variant of this approach called **hard negative mining**. This applies to situations where we have a moderate supply of positive examples, but an immense number of negative examples. In this case we can't use all the negative examples in training, but we need to search for negative examples that are most likely to improve the classifier's performance. We can do so by selecting a set of negative examples, training with these, and then searching the rest of the negative examples to find ones that generate false positives—these are hard negatives. We can iterate the procedure of training and searching for hard negatives; typically, we expand the pool of negative examples at each iteration.

Most basic classifier training algorithms expect that (a) the percentage of positive and negative examples in the training data set accurately reflects the test data and (b) that the cost of a false positive is the same as the cost of a false negative. On occasion, you will encounter data where one or the other assumption is not true. As an example, if you have one training positive example for every 10, 000 negative training examples, classifying everything as negative is a very good rule *if* the test data reflects those frequencies. But doing so may not be particularly useful, either because the test data doesn't have those frequencies — maybe you had a hard time finding training examples, or were lazy — or because the cost of a false negative is very high. In such cases you can **reweight** the data. For example, you might count each positive example 10 times rather than once when you compute the training loss. Most classifier codes have mechanisms to allow this. Choosing a weighting can be tricky, but it is usual to try several different weightings, then evaluate on a validation set to see which produces the most satisfactory results.

14.5.2 Building Multi-Class Classifiers Out of Binary Classifiers

There are two standard methods to build multi-class classifiers out of binary classifiers. In the **all-vs-all** approach, we train a binary classifier for each pair of classes. To classify an example, we present it to each of these classifiers. Each classifier decides which of two classes the example belongs to, then records a vote for that class. The example gets the class label with the most votes. This approach is simple, but scales very badly with the number of classes.

In the **one-vs-all** approach, we build a binary classifier for each class. This classifier must distinguish its class from all the other classes. We then take the class with the largest classifier score. One possible concern with this method is that training algorithms usually do not compel classifiers to be good at ranking examples. We train classifiers so that they give positive scores for positive examples, and negative scores for negative examples, but we do nothing explicit to ensure that a more positive score means the example is more like the positive class. Another

important concern is that the classifier scores must be calibrated to one another, so that when one classifier gives a larger positive score than another, we can be sure that the first classifier is more certain than the second. Some classifiers, such as logistic regression, report posterior probabilities, which require no calibration. Others, such as the SVM, report numbers with no obvious semantics and need to be calibrated. The usual method to calibrate these numbers is an algorithm due to **?**, which uses logistic regression to fit a simple probability model to SVM outputs. One-vs-all methods tend to be reliable and effective even when applied to uncalibrated classifier outputs, most likely because training algorithms do tend to encourage classifiers to rank examples correctly.

Neither strategy is particularly attractive when the number of classes is large, because the number of classifiers we must train scales poorly (linearly in one case, quadratically in the other) with the number of classes. If we were to allocate each class a distinct binary vector, we would need only $\log N$ bits in the vector for N classes. We could then train one classifier for each bit, and we should be able to classify into N classes with only $\log N$ classifiers. This strategy tends to founder on questions of which class should get which bit string, because this choice has significant effects on the ease of training the classifiers. Nonetheless, it gives an argument that suggests that we should not need as many as N classifiers to tell N classes apart.

14.5.3 Class Confusion Matrices

Evaluating a multi-class classifier is more complex than evaluating a binary classifier. There are only two kinds of mistake that a binary classifier can make. A multi-class classifier can make many more (mapping any one class to any other class). It is useful to know the total error rate of the classifier, which is the percentage of classification attempts that produce the wrong answer. An alternative is the accuracy, which is the percentage of classification attempts that produce the *right* answer. If the error rate is low enough, or the accuracy is high enough, there's not much to worry about. But if it's not, you can look at the **class confusion matrix** to see what's going on.

	Predict 0	Predict 1	Predict 2	Predict 3	Predict 4	Class error
True 0	151	7	2	3	1	7.9%
True 1	32	5	9	9	0	91%
True 2	10	9	7	9	1	81%
True 3	6	13	9	5	2	86%
True 4	2	3	2	6	0	100%

TABLE 14.1: The class confusion matrix for a multiclass classifier. Further details about the dataset and this example appear in worked example 1.

Table 14.1 gives an example. This is a class confusion matrix from a classifier built on a dataset where one tries to predict the degree of heart disease from a collection of physiological and physical measurements. There are five classes ($0 \ldots 4$). The i, j'th cell of the table shows the number of data points of true class i that

were classified to have class j. As I find it hard to recall whether rows or columns represent true or predicted classes, I have marked this on the table. For each row, there is a class error rate, which is the percentage of data points of that class that were misclassified. The first thing to look at in a table like this is the diagonal; if the largest values appear there, then the classifier is working well. This clearly isn't what is happening for table 14.1. Instead, you can see that the method is very good at telling whether a data point is in class 0 (the class error rate is rather small), but cannot distinguish between the other classes. This is a strong hint that the data can't be used to draw the distinctions that we want. It might be a lot better to work with a different set of classes.

14.5.4 Software for SVM's

We obtain a support vector machine by solving one of the constrained optimization problems given above. These problems have quite special structure, and one would usually use one of the many packages available on the web for SVMs to solve them.

LIBSVM (which can be found using Google, or at `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`) is a dual solver that is now widely used; it searches for nonzero Lagrange multipliers using a clever procedure known as SMO (sequential minimal optimization). A good primal solver is PEGASOS; source code can be found using Google, or at `http://www.cs.huji.ac.il/~shais/code/index.html`.

SVMLight (Google, or `http://svmlight.joachims.org/`) is a comprehensive SVM package with numerous features. It can produce sophisticated estimates of the error rate, learn to rank as well as to classify, and copes with hundreds of thousands of examples. Andrea Vedaldi, Manik Varma, Varun Gulshan, and Andrew Zisserman publish code for a multiple kernel learning-based image classifier at `http://www.robots.ox.ac.uk/~vgg/software/MKL/`. Manik Varma publishes code for general multiple-kernel learning at `http://research.microsoft.com/en-us/um/people/manik/code/GMKL/download.html`, and for multiple-kernel learning using SMO at `http://research.microsoft.com/en-us/um/people/manik/code/SMO-MKL/download.html`. Peter Gehler and Sebastian Nowozin publish code for their recent multiple-kernel learning method at `http://www.vision.ee.ethz.ch/~pgehler/projects/iccv09/index.html`.

14.6 WHAT YOU SHOULD REMEMBER - NEED SOMETHING

CHAPTER 15

Exploiting your Neighbors

Classification and regression problems are similar, because in each case we are trying to predict a value from a vector of features. For classification, we want to predict a label, and for regression we wanted to predict a number (we will do more shortly). Write y for what is predicted, and $\mathbf{x}$ for the vector of features. Now assume we have two examples, $\mathbf{x}_i$ and $\mathbf{x}_j$. We expect that, if $\mathbf{x}_i$ and $\mathbf{x}_j$ are close, the correct values of y_i and y_j will be close, too. What this means depends somewhat on the problem. If we are classifying the examples, then I mean that the labels should be the same for nearby examples; if we are regressing something against the features, we expect the values of the dependent variable to be close, too.

This observation suggests the extremely useful and general strategy of exploiting a data item's neighbors. If you want to classify a data item, find the closest example, and report the class of that example. Alternatively, you could find the closest k examples, and vote. Similarly, if you want to predict something using the data item, find the closest k examples, and construct a prediction using them.

This strategy has many very attractive features. It turns out to produce accurate classifiers, and it is easy to build multi-class classifiers like this (Section **??**). It is easy to regress numbers against features like this (Section **??**); but it is also easy to regress complex objects against features, too (Section **??**). However, one must find the nearest neighbors, which can be challenging (Section **??**).

15.1 CLASSIFYING WITH NEAREST NEIGHBORS

Imagine we have a data point $\mathbf{x}$ that we wish to classify (a query point). Our strategy will be to find the closest training example, and report its class.

How well can we expect this strategy to work? Remember that any classifier will slice up the space of examples into cells (which might be quite complicated) where every point in a cell has the same label. The boundaries between cells are decision boundaries — when a point passes over the decision boundary, the label changes. Now assume we have a large number of labelled training examples, and we know the best possible set of decision boundaries. If there are many training examples, there should be at least one training example that is close to the query point. If there are enough training examples, then the closest point should be inside the same cell as the query point.

You may be worried that, if the query point is close to a decision boundary, the closest point might be on the other side of that boundary. But if it were, we could improve things by simply having more training points. All this suggests that, *with enough training points*, our classifier should work about as well as the best possible classifier. This intuition turns out to be correct, though the number of training points required is wholly impractical, particularly for high-dimensional feature vectors.

One important generalization is to find the k nearest neighbors, then choose

a label from those. A (k, l) nearest neighbor classifier finds the k example points closest to the point being considered, and classifies this point with the class that has the highest number of votes, as long as this class has more than l votes (otherwise, the point is classified as unknown). A $(k, 0)$-nearest neighbor classifier is usually known as a
k-nearest neighbor classifier, and a $(1, 0)$-nearest neighbor classifier is usually known as a **nearest neighbor classifier**. In practice, one seldom uses more than three nearest neighbors. Finding the k nearest points for a particular query can be difficult, and Section 15.4.1 reviews this point.

There are three practical difficulties in building nearest neighbor classifiers. You need a lot of labelled examples. You need to be able to find the nearest neighbors for your query point. And you need to use a sensible choice of distance. For features that are obviously of the same type, such as lengths, the usual metric may be good enough. But what if one feature is a length, one is a color, and one is an angle? One possibility is to whiten the features (section 1). This may be hard if the dimension is so large that the covariance matrix is hard to estimate. It is almost always a good idea to scale each feature independently so that the variance of each feature is the same, or at least consistent; this prevents features with very large scales dominating those with very small scales. Notice that nearest neighbors (fairly obviously) doesn't like categorical data. If you can't give a clear account of how far apart two things are, you shouldn't be doing nearest neighbors. It is possible to fudge this point a little, by (say) putting together a distance between the levels of each factor, but it's probably unwise.

Nearest neighbors is wonderfully flexible about the labels the classifier predicts. Nothing changes when you go from a two-class classifier to a multi-class classifier. In fact, you could attach a number (or more) to each point and predict that as well (Section **??**).

Worked example 15.1 *Classifying using nearest neighbors*

Build a nearest neighbor classifier to classify the digit data originally constructed by Yann Lecun. You can find it at several places. The original dataset is at `http://yann.lecun.com/exdb/mnist/`. The version I used was used for a Kaggle competition (so I didn't have to decompress Lecun's original format). I found it at `http://www.kaggle.com/c/digit-recognizer`.

Solution: As you'd expect, R has nearest neighbor code that seems quite good (I haven't had any real problems with it, at least). There isn't really all that much to say about the code. I used the R FNN package. This uses a bagging strategy. There is sample code in listing **??**. I trained on 1000 of the 42000 examples, so you could see how in the code. I tested on the next 200 examples. For this (rather small) case, I found the following class confusion matrix

	P									
	0	1	2	3	4	5	6	7	8	9
0	12	0	0	0	0	0	0	0	0	0
1	0	20	4	1	0	1	0	2	2	1
2	0	0	20	1	0	0	0	0	0	0
3	0	0	0	12	0	0	0	0	4	0
4	0	0	0	0	18	0	0	0	1	1
5	0	0	0	0	0	19	0	0	1	0
6	1	0	0	0	0	0	18	0	0	0
7	0	0	1	0	0	0	0	19	0	2
8	0	0	1	0	0	0	0	0	16	0
9	0	0	0	2	3	1	0	1	1	14

There are no class error rates here, because I was in a rush and couldn't recall the magic line of R to get them. However, you can see the classifier works rather well for this case.

15.2 EXPLOITING YOUR NEIGHBORS TO PREDICT A NUMBER

Nearest neighbors can clearly predict a number for a query example — you find the closest training example, and report its number. This would be one way to use nearest neighbors for regression, but it isn't terribly effective. One important difficulty is that the regression prediction is piecewise constant (Figure 15.1). If there is an immense amount of data, this may not present major problems, because the steps in the prediction will be small and close together. But it's not generally an effective use of data.

A more effective strategy is to find several nearby training examples, and use them to produce an estimate. This approach can produce very good regression estimates, because every prediction is made by training examples that are near to the query example. However, producing a regression estimate is expensive, because for every query one must find the nearby training examples.

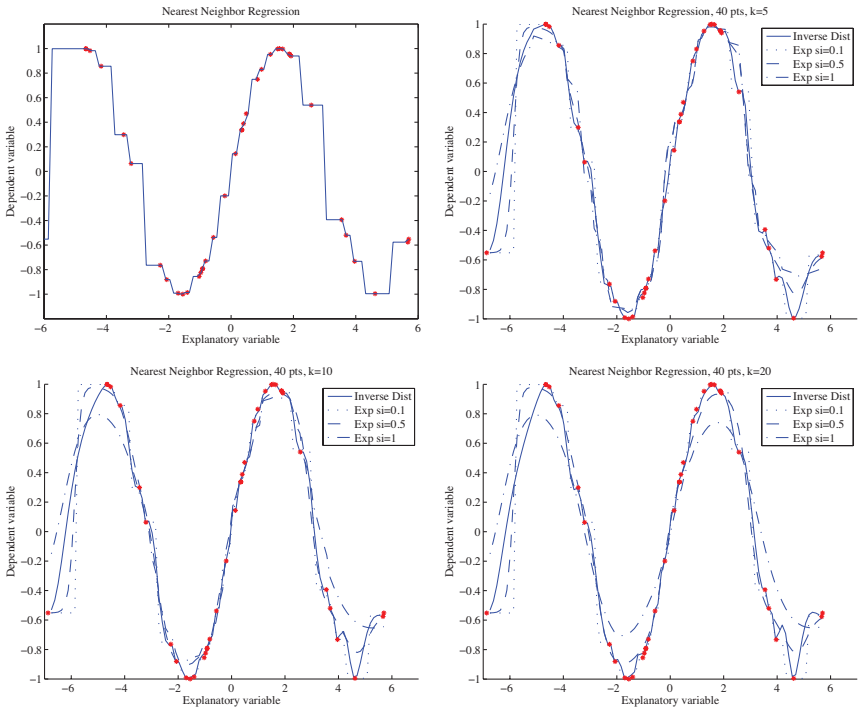

FIGURE 15.1: *Different forms of nearest neighbors regression, predicting y from a one-dimensional x, using a total of 40 training points.* **Top left:** *reporting the nearest neighbor leads to a piecewise constant function.* **Top right:** *improvements are available by forming a weighted average of the five nearest neighbors, using inverse distance weighting or exponential weighting with three different scales. Notice if the scale is small, then the regression looks a lot like nearest neighbors, and if it is too large, all the weights in the average are nearly the same (which leads to a piecewise constant structure in the regression).* **Bottom left** *and* **bottom right** *show that using more neighbors leads to a smoother regression.*

Write $\mathbf{x}$ for the query point, and assume that we have already collected the N nearest neighbors, which we write $\mathbf{x}_i$. Write y_i for the value of the dependent variable for the i'th of these points. Notice that some of these neighbors could be quite far from the query point. We don't want distant points to make as much contribution to the model as nearby points. This suggests forming a weighted average of the predictions of each point. Write w_i for the weight at the i'th point. Then the estimate is

$$y_{pred} = \frac{\sum_i w_i y_i}{\sum_i w_i}.$$

A variety of weightings are reasonable choices. Write $d_i = \|(\mathbf{x} - \mathbf{x}_i)\|$ for the distance between the query point and the i'th nearest neighbor. Then inverse distance weighting uses $w_i = 1/d_i$. Alternatively, we could use an exponential

function to strongly weight down more distant points, using

$$w_i = \exp\left(\frac{-d_i^2}{2\sigma^2}\right).$$

We will need to choose a scale σ, which can be done by cross-validation. Hold out some examples, make predictions at the held out examples using a variety of different scales, and choose the scale that gives the best held-out error. Alternatively, if there are enough nearest neighbors, we could form a distance weighted linear regression, then predict the value at the query point from that regression.

Each of these strategies presents some difficulties when $\mathbf{x}$ has high dimension. In that case, it is usual that the nearest neighbor is a lot closer than the second nearest neighbor. If this happens, then each of these weighted averages will boil down to evaluating the dependent variable at the nearest neighbor (because all the others will have very small weight in the average).

15.3 REGRESSING COMPLEX OBJECTS

In the linear regression model, I did not explain how to regress objects with complicated structure against explanatory variables. This is a problem that arises often in practice. For example, I might want to predict a parse tree (the object with complicated structure) from a sentence (the explanatory variables). As another example, I might want to predict a map of the shadows in an image (the object with complicated structure) against an image (the explanatory variables). As yet another example, I might want to predict which direction to move the controls on a radio-controlled helicopter (the object with complicated structure) against a path plan and the current state of the helicopter (the explanatory variables). In each case, it seems unlikely that a pure linear regression is likely to resolve the problem.

Looking at neighbors is a very good way to solve such problems. The general strategy is relatively simple. We find a large collection of pairs of training data. Write $\mathbf{x}_i$ for the explanatory variables for the i'th example, and $\mathbf{y}_i$ for the dependent variable in the i'th example. This dependent variable could be anything — it doesn't need to be a single number. It might be a tree, or a shadow map, or a word, or anything at all. I wrote it as a vector because I needed to choose some notation.

In the simplest, and most general, approach, we obtain a prediction for a new set of explanatory variables $\mathbf{x}$ by (a) finding the nearest neighbor and then (b) producing the dependent variable for that neighbor. We might vary the strategy slightly by using an approximate nearest neighbor. You have already seen this strategy in action filling holes in images.

If the dependent variables have enough structure that it is possible to summarize a collection of different dependent variables, then we might recover the k nearest neighbors and summarize their dependent variables. How we summarize rather depends on the dependent variables. For example, it is a bit difficult to imagine the average of a set of trees, but quite straightforward to average numbers. If the dependent variable was a word, we might not be able to average words, but we can vote and choose the most popular word. If the dependent variable is a vector, we can compute either distance weighted averages or a distance weighted

linear regression.

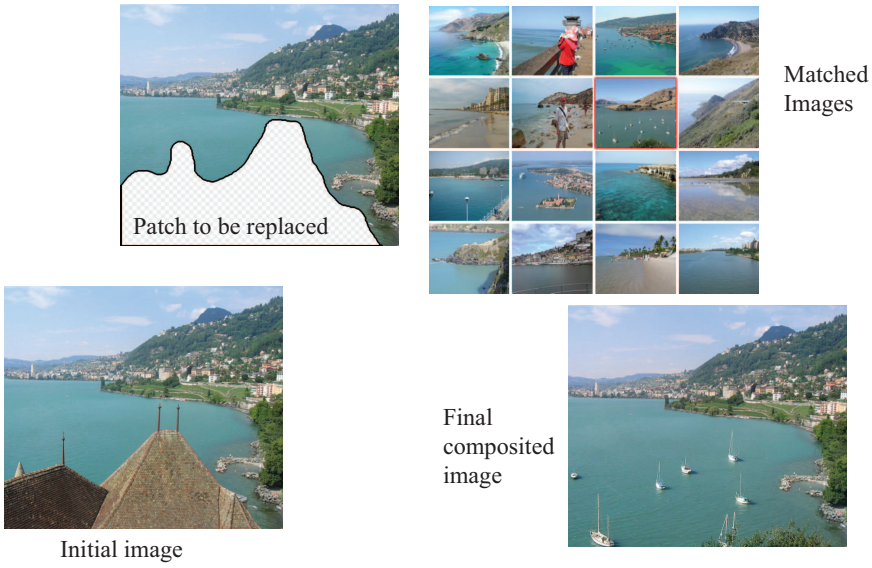

FIGURE 15.2: *We can fill large holes in images by matching the image to a collection, choosing one element of the collection, then cutting out an appropriate block of pixels and putting them into the hole in the query image. In this case, the hole has been made by an artist, who wishes to remove the roofline from the view. Notice how there are a range of objects (boats, water) that have been inserted into the hole. These objects are a reasonable choice, because the overall structures of the query and matched image are largely similar — getting an image that matches most of the query image supplies enough context to ensure that the rest "makes sense".*

15.3.1 Example: Filling Large Holes with Whole Images

Many different kinds of user want to remove things from images or from video. Art directors might like to remove unattractive telephone wires; restorers might want to remove scratches or marks; there's a long history of government officials removing people with embarrassing politics from publicity pictures (see the fascinating examples in **?**); and home users might wish to remove a relative they dislike from a family picture. All these users must then find something to put in place of the pixels that were removed.

If one has a large hole in a large image, we may not be able to just extend a texture to fill the hole. Instead, entire objects might need to appear in the hole (Figure 15.2). There is a straightforward, and extremely effective, way to achieve this. We match the image to a large collection of images, to find the nearest neighbors (the details of the distance function are below). This yields a set of example images where all the pixels we didn't want to replace are close to those of the query image. From these, we choose one, and fill in the pixels from that image.

There are several ways to choose. If we wish to do so automatically, we could

use the example with the smallest distance to the image. Very often, an artist is involved, and then we could prepare a series of alternatives — using, perhaps, the k closest examples — then show them to the artist, who will choose one. This method, which is very simple to describe, is extremely effective in practice.

It is straightforward to get a useful distance between images. We have an image with some missing pixels, and we wish to find nearby images. We will assume that all images are the same size. If this isn't in fact the case, we could either crop or resize the example images. A good measure of similarity between two images $\mathcal{A}$ and $\mathcal{B}$ can be measured by forming the **sum of squared differences** (or **SSD**) of corresponding pixel values. You should think of an image as an array of pixels. If the images are grey-level images, then each pixel contains a single number, encoding the grey-level. If the images are color images, then each pixel (usually!) contains three values, one encoding the red-level, one encoding the green-level, and one encoding the blue-level. The SSD is computed as

$$\sum_{(i,j)} (\mathcal{A}_{ij} - \mathcal{B}_{ij})^2$$

where i and j range over all pixels. If the images are grey-level images, then by $(\mathcal{A}_{ij} - \mathcal{B}_{ij})^2$, I mean the squared difference between grey levels; if they are color images, then this means the sum of squared differences between red, green and blue values. This distance is small when the images are similar, and large when they are different (it is essentially the length of the difference vector).

Now we don't know some of the pixels in the query image. Write $\mathcal{K}$ for the set of pixels around a point whose values are known, and $\sharp\mathcal{K}$ for the size of this set. We can now use

$$\frac{1}{\sharp\mathcal{K}} \sum_{(i,j)\in\mathcal{K}} (\mathcal{A}_{ij} - \mathcal{B}_{ij})^2.$$

Filling in the pixels requires some care. One does not usually get the best results by just copying the missing pixels from the matched image into the hole. Instead, it is better to look for a good **seam**. We search for a curve enclosing the missing pixels which (a) is reasonably close to the boundary of the missing pixels and (b) gives a good boundary between the two images. A good boundary is one where the query image (on one side) is "similar to" the matched image (on the other side). A good sense of similarity requires that pixels match well, and that image gradients crossing the boundary tend to match too.

15.3.2 Filling in a Single Missing Pixel

Imagine I have an image where exactly one pixel value is missing. This pixel could have been erased by noise; it could be a tiny scratch on the film of the image; the disk storage system might have lost it; and so on. I would like to create a value to insert into that pixel's location.

Here is an easy strategy. Take a block of pixels centered on the missing pixel. Find many close matches to this block of pixels in a collection of images, counting only the known pixel values. Each match offers a possible value for the missing pixel. There is no reason to prefer any one value, so choose randomly from the values.

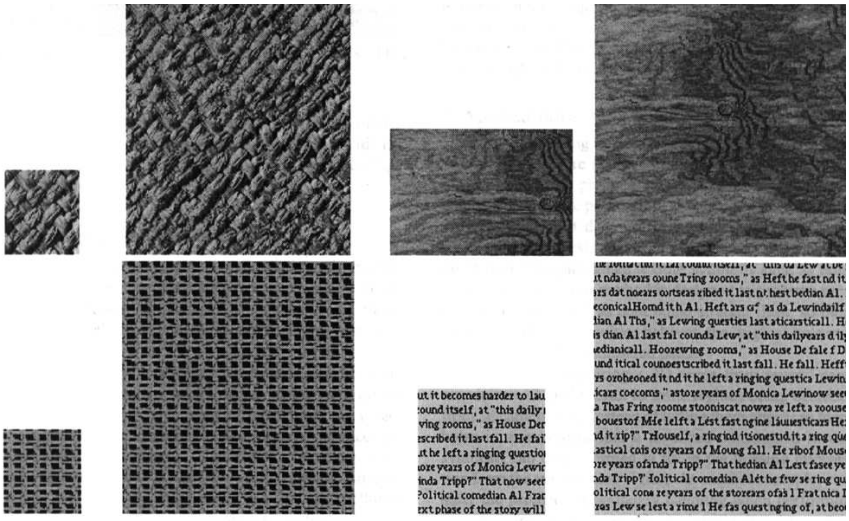

FIGURE 15.3: *? synthesize textures by matching neighborhoods of the image being synthesized to the example image, and then choosing at random amongst the possible values reported by matching neighborhoods (Algorithm 15.1). This means that the algorithm can reproduce complex spatial structures, as these examples indicate. The small block on the* **left** *is the example texture; the algorithm synthesizes the block on the* **right***. Note that the synthesized text looks like text: it appears to be constructed of words of varying lengths that are spaced like text, and each word looks as though it is composed of letters (though this illusion fails as one looks closely).* This figure was originally published as Figure 3 of "Texture Synthesis by Non-parametric Sampling," A. Efros and T.K. Leung, Proc. IEEE ICCV, 1999 © IEEE, 1999.

We could use the SSD to match these patches, but this measure places the same weight on pixels close to the unknown value as it does on distant pixels. Better results are usually obtained by weighting up nearby pixels and weighting down distant pixels. Now, for convenience, we will assume the patch is square, and the missing pixel value has index $(0, 0)$. The function

$$\exp\left(\frac{-(i^2 + j^2)}{2\sigma^2}\right)$$

takes the value one when $(i, j) = (0, 0)$, and falls off fast as $(i^2 + j^2)$ — distance from the origin — grows. If we weight squared pixel differences by this function, then the difference between pixels far from the missing pixel is less important in determining the value of the missing pixels. This yields **Gaussian weighted SSD**, where the matching cost is

$$\sum_{(i,j)\in\text{patch},(i,j)\neq(0,0)} (\mathcal{A}_{ij} - \mathcal{B}_{ij})^2 \exp\left(\frac{-(i^2 + j^2)}{2\sigma^2}\right).$$

Patches with a smaller value of this match cost are better matches.

If we need to fill in only one pixel, then finding the best-matching patch is likely enough. However, we can generalize the idea to fill in more than one pixel. When we do so, it will be important to preserve random structures in the image textures. This means it is a good idea to collect several alternative values for the pixel, and choose one at random. To do so, we find the closest patch. Write s_{min} for the similarity value of the closest patch. We then recover all patches whose similarity value is less than $(1 + \epsilon)s_{min}$. Notice that doing so may involve a linear pass through the data (which should be slow, because we should be using a lot of data) or an approximate nearest neighbors query.

We may not need to search a large collection of images to find good patches. Most image textures occupy reasonably large regions in the image, meaning we might find quite good patches matching our patch in the original image itself.

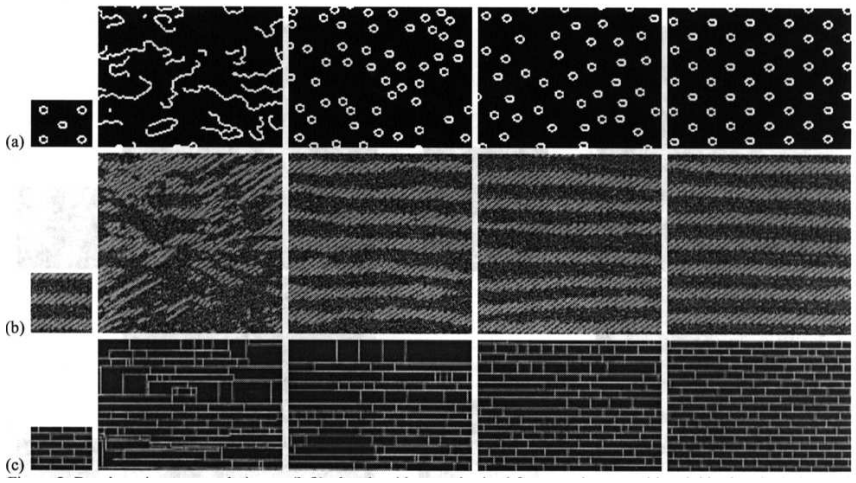

FIGURE 15.4: *The size of the image neighborhood to be matched makes a significant difference in Algorithm 15.1. In the figure, the textures at the right are synthesized from the small blocks on the* **left**, *using neighborhoods that are increasingly large as one moves to the* **right**. *If very small neighborhoods are matched, then the algorithm cannot capture large-scale effects easily. For example, in the case of the spotty texture, if the neighborhood is too small to capture the spot structure (and so sees only pieces of curve), the algorithm synthesizes a texture consisting of curve segments. As the neighborhood gets larger, the algorithm can capture the spot structure, but not the even spacing. With very large neighborhoods, the spacing is captured as well.* This figure was originally published as Figure 2 of "Texture Synthesis by Non-parametric Sampling," A. Efros and T.K. Leung, Proc. IEEE ICCV, 1999 © IEEE, 1999.

15.3.3 Filling in a Textured Region

Now imagine we have many missing pixels. If each is scattered widely across the image, the original procedure will work fine, because each missing pixel will be

spanned by a patch. But things get more interesting if we are missing a block of pixels. In this case, the values of some pixels in the neighborhood of the pixel to be synthesized are not known; these pixels need to be synthesized too. We assume that the missing pixels are inside a region of coherent texture, so (for example) we might be expected to create a region of grass, or of bushes, or of sea. We don't expect to need to create objects (boats at sea; birds in bushes; and so on). This means we don't need to worry about whether the objects "naturally belong" in the picture.

We can proceed as follows. Choose a pixel to synthesize. Find a set of examples for the pixel of interest, by matching a patch of image around that pixel to some example images (quite possibly other parts of the original image). We will use SSD with Gaussian weights to match, but omit the values of the pixels we don't know (as in Section 15.3.1). Notice the cost there is weighted by the number of known pixels; this is important, because different query patches might have different numbers of known pixels.

One important application of this procedure is **texture synthesis**, where one takes a small block of texture and expands it to a large block. This is useful, because computer graphics methods and computer games are aggressive consumers of textures, and want to be supplied with very large texture regions. But large examples of sample textures are hard to get — we want to build a large texture from a small one. We could do so by regarding the small texture as part of a larger image, with many missing pixels, yielding Algorithm 15.1.

The size and shape of the neighborhood we use is significant, because it codes the range over which pixels can affect one another's values directly (see Figure 15.4). The original method, due to Efros *et al.* used a square neighborhood, centered at the pixel of interest; current methods do so as well.

```
Choose a texture sample
Insert this sample into the image to be synthesized
Until each location in the image to be synthesized has a value
  For each unsynthesized location on
    the boundary of the block of synthesized values
    Match the neighborhood of this location to the
      example image, ignoring unsynthesized
      locations in computing the matching score
    Choose a value for this location uniformly and at random
      from the set of values of the corresponding locations in the
      matching neighborhoods
  end
end
```

Algorithm 15.1: *Non-parametric Texture Synthesis.*

15.4 FINDING YOUR NEAREST NEIGHBORS

15.4.1 Finding the Nearest Neighbors and Hashing

To build a nearest neighbors classifier, we need to find the members of a set of high dimensional vectors that are closest to some query vector. It turns out this is a general, and quite difficult, problem. A linear search through the dataset is fine for a small set of data items, but we will operate at scales where we need something more efficient. Surprisingly, exact solutions turn out to be only very slightly more efficient than a linear search through the dataset. Approximate solutions are much more efficient. They are approximate in the sense that, with high probability, they return a point that is almost as close to the query as the closest point. The main trick to obtaining a good approximate solution is to carve the space into cells, then look at items that lie in cells near the query vector; there are two methods that are worth discussing in detail here.

Locality Sensitive Hashing

In **locality sensitive hashing**, we build a set of hash tables containing the data items, using different hashing functions for each table. For a query item, we recover whatever is in each hash table at the location corresponding to the hash code computed for the query item. We search this set, keeping any data items from this set that are sufficiently close to the query. There are many choices of hash function; the most widely used in vision is **random projection**. Write $\mathbf{v}$ for a vector, representing either a query or a data item. We now obtain a single bit of a hash code by choosing a random vector $\mathbf{r}$ and then computing $sign(\mathbf{v} \cdot \mathbf{r})$. Computing a k-bit hash code involves choosing k such random vectors, then computing one bit for each. There is a set of k such random vectors associated with each hash table. Geometrically, choosing an $\mathbf{r}$ corresponds to choosing a hyperplane in the data space, and the hashing bit corresponds to which side of the hyperplane $\mathbf{v}$ lies on. A k-bit hash code identifies a cell in an arrangement of k hyperplanes in which $\mathbf{v}$ lies. k will be small compared to the dimension, and so we are cutting the space into 2^k cells. This means that there will be relatively few data items that lie in the same cell as a query. Some nearby data items may not lie in the same cell, because they could be on the other side of a hyperplane, but these items should lie in the same cell in another hash table.

All these assertions can be made precise, resulting in a guarantee that: (a) a data item that is almost as close as the nearest neighbor will be found with high probability; and (b) all data items closer to the query than some threshold will be found with high probability, whereas data items that are significantly more distant will be found with low probability. Straightforward geometric intuition suggests that this approach will work best when the data items have zero mean, which is easy to arrange. Notice that using n k-bit hash tables is not the same as using one nk-bit hash table. In the first case, the list of points returned from a particular query is a union of the lists returned from each of the n hash tables. This means that points that are near the query but just happen to lie outside the query's cell for one hash table, have a good chance of being found in another hash table. In the second case, the list we must handle is much shorter (because there are more cells), but there is a better chance of missing nearby points. The choice of n and k will

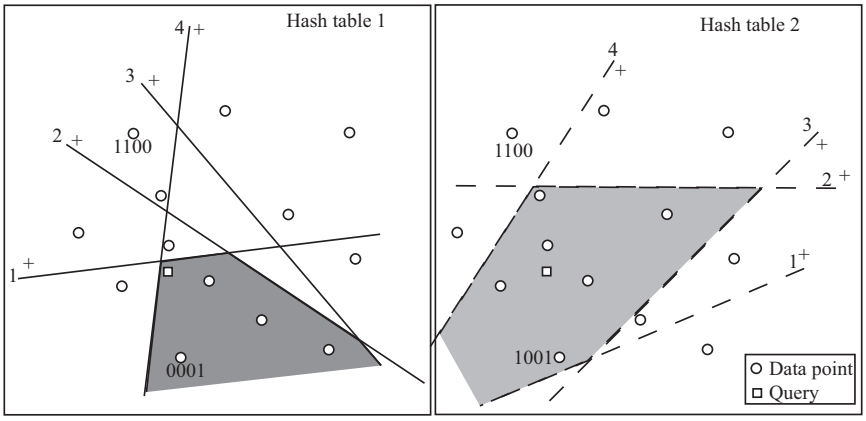

FIGURE 15.5: *In locality sensitive hashing using a random projection hash function, the hash function is equivalent to a hyperplane in the data space. Items that lie on one side of the hyperplane corresponding to the nth bit have that bit set to one; otherwise, it is zero. These hyperplanes cut the space of data into a set of cells. All the data items in a cell get a binary hash code (shown for two points in each figure; we have marked the order of the bits by labeling the hyperplanes, and the +s show which side of the hyperplane gets a one). To query, we find all data items in the same hash table entry as the query (the filled polygons in the figure), and then find the closest. However, the nearest neighbor might not be in this cell (for example, the case on the* **left***). To reduce the probability of error from this cause, we use more than one hash table and search the union of the sets of points lying in the query cell. In the case illustrated, the nearest neighbor of the query lies in the query cell for the second hash table, on the* **right***. The hash tables reduce the set of points we need to search, with high probability of finding a point that is almost as close as the nearest neighbor.*

depend on dimension and on the nature of the probabilistic guarantee one wants. There are a variety of other possible choices of hash function. Details of other choices, and precise statements of the relevant guarantees, can be found in (**?**).

KD-Trees for Approximate Nearest Neighbors

Random projection methods build a cell structure that is independent of the distribution of the data. This means trouble if data is heavily concentrated in some regions, because queries that land in a heavily populated cell of the hash table will need to search a long list. An alternative method is to use a **k-d tree** to build the cell structure. A k-d tree is built by recursively splitting cells. The root will be the whole space. To generate the children of a cell, select one dimension d, perhaps at random, and select some threshold value t_d. Write the dth component of $\mathbf{v}$ as v_d. Now all data items in a cell with $v_d \leq t_d$ are placed in the left child, and all others

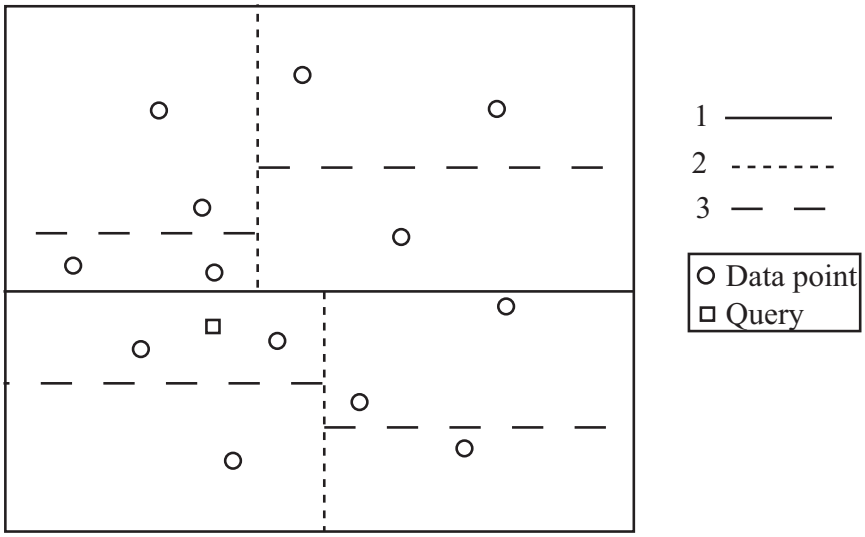

FIGURE 15.6: *A k-d tree is built by recursively splitting cells along dimensions. The order in which cells are split for this tree is shown by the dashes on the lines. The nearest neighbor for the query point is found by (a) finding the closest item in the query point's cell, then (b) backtracking, and looking at cells that could contain closer items. Notice that in this example one will need to go right up to the root of the tree and down the other side to find the nearest neighbor. In high dimensions, this backtracking becomes intractable, but if an approximate nearest neighbor is sufficient, the amount of backtracking can be controlled successfully.*

in the right. We now apply this splitting procedure recursively to the root, until the children are sufficiently small. If we choose the threshold value appropriately (for example, the median of the data in the cell), we can ensure that cells are small in dense components of the space and large in sparse components.

The nearest neighbor to a query can then be found by walking the tree to find the cell containing the query point. We then check any data items in that cell. Write the distance from the query to the closest as d_c. We now backtrack, investigating cells that could contain points closer than d_c and updating d_c when we find a better point. We can prune any branch of the tree that represents a volume that is further from the query than d_c. This procedure works well for low dimensions, but becomes unattractive in high dimensions because we will need to explore too many cells (the number of neighbors of a cell goes up exponentially with dimension).

This difficulty can be avoided if an approximate nearest neighbor is sufficient. In the **best bin first** approach, we look at a fixed number N_c of cells, then report the best point found so far. Promising cells will tend to have some points that are close to the query, and we define the distance between a cell and the query to be the shortest distance from the query to any point on the cell's boundary. Whenever we investigate the child of a cell, we insert the other child into a priority queue,

ordered by distance to the query. Once we have checked a cell, we retrieve the next cell from the priority queue. We do this until we have looked at N_c cells. We will look mainly at cells that are close to the query, and so the point we report is a good approximate nearest neighbor.

Good performance of a particular method depends somewhat on the dataset. For most applications, the choice of method can be made offline using the dataset, or a subset of it. **?** describe a software package that can choose an approximate nearest neighbors method that is fastest for a particular dataset. Generally, they find that using multiple randomized k-d trees is usually the best; at the time of writing, software could be found at `http://www.cs.ubc.ca/~mariusm/index.php/FLANN/FLANN`.

15.5 WHAT YOU SHOULD REMEMBER - NEED SOMETHING

CHAPTER 16

Describing Repetition with Vector Quantization

Classifying images is an important problem, with a variety of applications. I list some below. We have seen powerful classification methods. But what features should one use to classify images? In this section, I will describe a construction that extends to a variety of other signals, because it exploits repetition. Repetition is an important feature of many interesting signals. For example, images contain *textures*, which are orderly patterns that look like large numbers of small structures that are repeated. Examples include the spots of animals such as leopards or cheetahs; the stripes of animals such as tigers or zebras; the patterns on bark, wood, and skin. Similarly, speech signals contain *phonemes* — characteristic, stylised sounds that people assemble together to produce speech (for example, the "ka" sound followed by the "tuh" sound leading to "cat"). An important part of representing signals that repeat is identifying the components that repeat, then describing the signal in terms of those components.

Repetition occurs in more subtle forms than spot patterns. The essence is that a small number of local patterns can be used to represent a large number of examples. You see this effect in pictures of scenes. If you collect many pictures of, say, a beach scene, you will expect most to contain some waves, some sky, and some sand. The individual patches of wave, sky or sand can be surprisingly similar, and different images are made by selecting some patches from a vocabulary of patches, then placing them down to form an image. Similarly, pictures of living rooms contain chair patches, TV patches, and carpet patches. Many different living rooms can be made from small vocabularies of patches; but you won't often see wave patches in living rooms, or carpet patches in beach scenes.

This view suggests that we should (a) develop a vocabulary of image patches and then (b) construct a feature that describes which vocabulary elements appear in the image. It turns out that where the vocabulary elements appear is less important. This is because different pictures of a living room may have the same patches in different locations (for example, you might just move the camera to the left). As another example, zebras have stripes, but precisely where the stripes lie on a zebra doesn't seem to matter much in practice.

The patches in the vocabulary are often referred to as **textons**. We could find these textons by looking for image patches that are common, and then reason about which ones are repeated. There are two important difficulties in finding image patches that commonly occur together. First, these representations of the image are continuous. We cannot simply count how many times a particular pattern occurs, because each vector is slightly different. Second, the representation is high dimensional in either case. A patch around a pixel might need hundreds of pixels

to represent it well. This means we cannot build a histogram directly, because it will have an unmanageable number of cells.

16.0.1 Applications of Image Classification

Detecting explicit images: There are numerous reasons to try and detect images that depict nudity or sexual content. In some jurisdictions, possession or distribution of such pictures might be illegal. Many employers want to ensure that work computers are not used to collect or view such images; in fact, vendors of image filtering software have tried to persuade employers that they might face litigation if they don't do so. Advertisers want to be careful that their ads appear only next to images that will not distress their customers. Web search companies would like to allow users to avoid seeing images they find distressing.

Material classification: Imagine we have an image window. What material (e.g., "wood," "glass," "rubber," and "leather") does the window cover? If we could answer this question, we could decide whether an image patch was cloth—and so might be part of a person, or of furniture—or grass, or trees, or sky. Generally, different materials produce different image textures, so natural features to use for this multiclass classification problem will be texture features. However, materials tend to have some surface relief (for example, the little pores on the surface of an orange; the bumps on stucco; the grooves in tree bark), and these generate quite complex shading behavior. Changes in the illumination direction can cause the shadows attached to the relief to change quite sharply, and so the overall texture changes. Furthermore, as Figure 16.1 illustrates, texture features might indicate the material an object is made of, but objects can have the same texture and be made of quite different materials.

Scene classification: Pictures of a bedroom, of a kitchen, or of a beach show different **scenes**. Scenes provide an important source of context to use in interpreting images. You could reasonably expect to see a pillow, but not a toaster or a beachball, in a bedroom; a toaster, but not a pillow or a beachball, in a kitchen; or a beachball and perhaps a pillow, but not a toaster, on a beach. We should like to be able to tell what scene is depicted in an image. This is difficult, because scenes vary quite widely in appearance, and this variation has a strong spatial component. For example, the toaster could be in many different locations in the kitchen. In **scene classification**, one must identify the scene depicted in the image. Scene labels are more arbitrary than object labels, because there is no clear consensus on what the different labels should be. Some labels seem fairly clear and are easy to assign accurately (Figure 16.2), for example, "kitchen," "bedroom," and other names of rooms in a house. Other labels are uncertain: should one distinguish between "woodland paths" and "meadows", or just label them all "outdoors"? Humans seem to have a problem here, too (Figure **??**). However, there are several scene datasets, each with its own set of labels, so methods can be compared and evaluated.

16.0.2 Vector Quantization and Textons

Vector quantization is a strategy to deal with these difficulties. Vector quantization is a way of representing vectors in a continuous space with numbers from a

FIGURE 16.1: *Material is not the same as object category (the three cars on the* **top** *are each made of different materials), and is not the same as texture (the three checkered objects on the* **bottom** *are made of different materials). Knowing the material that makes up an object gives us a useful description, somewhat distinct from its identity and its texture.* This figure was originally published as Figures 2 and 3 of "Exploring Features in a Bayesian Framework for Material Recognition," by C. Liu, L. Sharan, E. Adelson, and R. Rosenholtz Proc. CVPR 2010, 2010 © IEEE, 2010.

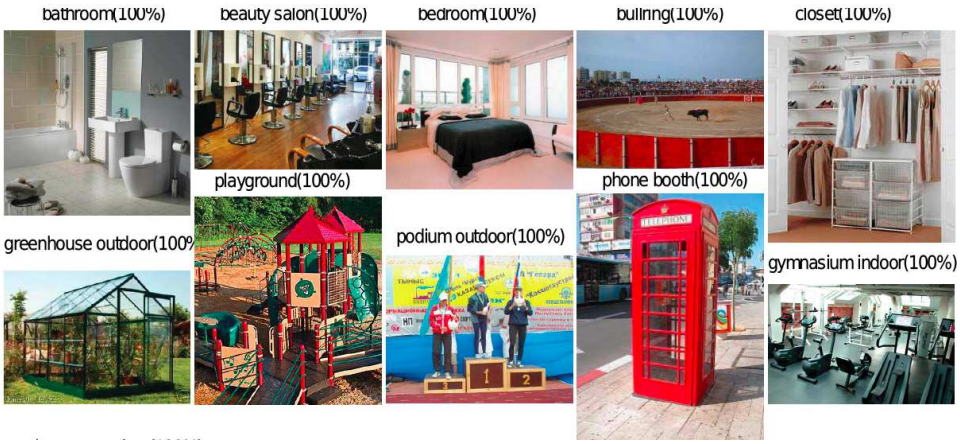

FIGURE 16.2: *Some scenes are easily identified by humans. These are examples from the SUN dataset (?) of scene categories that people identify accurately from images; the label above each image gives its scene type.* This figure was originally published as Figure 2 of "SUN database: Large-scale Scene Recognition from Abbey to Zoo," by J. Xiao, J. Hays, K. Ehinger, A. Oliva, and A. Torralba, Proc. IEEE CVPR 2010, © IEEE, 2010.

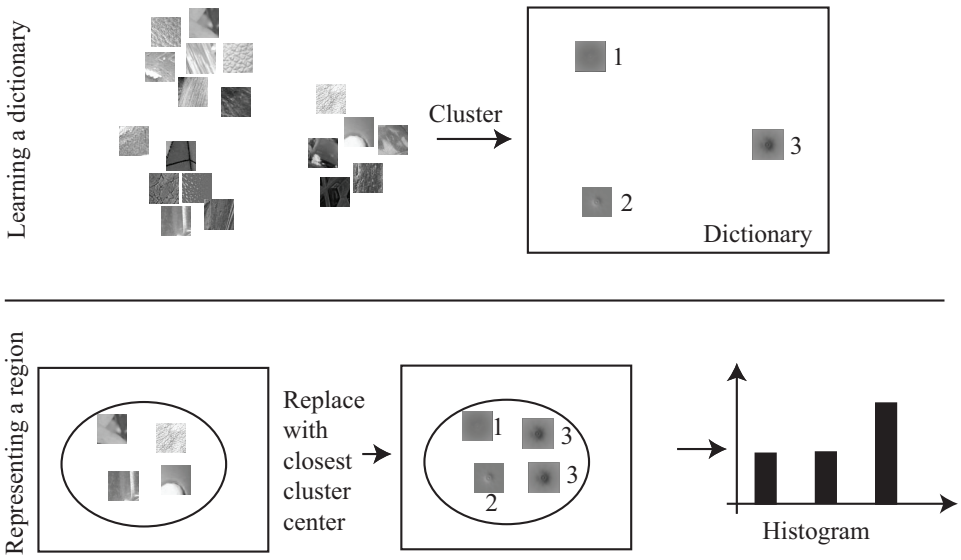

FIGURE 16.3: *There are two steps to building a pooled texture representation for a texture in an image domain. First, one builds a dictionary representing the range of possible pattern elements, using a large number of texture patches. This is usually done in advance, using a training data set of some form. Second, one takes the patches inside the domain, vector quantizes them by identifying the number of the closest cluster center, then computes a histogram of the different cluster center numbers that occur within a region. This histogram might appear to contain no spatial information, but this is a misperception. Some frequent elements in the histogram are likely to be textons, but others describe common ways in which textons lie close to one another; this is a rough spatial cue.* This figure shows elements of a database collected by C. Liu, L. Sharan, E. Adelson, and R. Rosenholtz, and published at `http://people.csail.mit.edu/lavanya/research_sharan.html`. Figure by kind permission of the collectors.

set of fixed size. We first build a set of clusters out of a training set of vectors; this set of clusters is often thought of as a dictionary. We now replace any new vector with the cluster center closest to that vector. This strategy applies to vectors quite generally, though we will use it for texture representation. Many different clusterers can be used for vector quantization, but it is most common to use k-means or one of its variants.

We can now represent a collection of vectors as a histogram of cluster centers. This general recipe can be applied to image representation by describing each pixel in the domain with some vector, then vector quantizing and describing the domain with the histogram of cluster centers. One natural vector to use is obtained by reshaping the pixels from a fixed-size patch around the image pixel (Figure 16.4). There is a rich collection of alternative possibilities in the computer vision literature.

Build a dictionary:
    Collect many training example images
    Construct the vectors $\mathbf{x}$ for relevant pixels; these could be
      a reshaping of a patch around the pixel, or a vector of filter outputs
      computed at the pixel, or other image representations.
    Obtain k cluster centers $\mathbf{c}$ for these examples

Represent an image domain:
    For each relevant pixel i in the image
        Compute the vector representation $\mathbf{x}_i$ of that pixel
        Obtain j, the index of the cluster center $\mathbf{c}_j$ closest to that pixel
        Insert j into a histogram for that domain

Algorithm 16.1: *Image Representation Using Vector Quantization.*

16.0.3 Hierarchical K Means

One important difficulty occurs in applications. We might need to have an enormous dataset (millions of image patches are a real possibility), and so a very large k. In this case, k means clustering becomes difficult because identifying which cluster center is closest to a particular data point scales linearly with k (and we have to do this for every data point at every iteration). There are two useful strategies for dealing with this problem.

The first is to notice that, if we can be reasonably confident that each cluster contains many data points, some of the data is redundant. We could randomly subsample the data, cluster that, then keep the cluster centers. This works, but doesn't scale particularly well.

A more effective strategy is to build a hierarchy of k-means clusters. We randomly subsample the data (typically, quite aggressively), then cluster this with a small value of k. Each data item is then allocated to the closest cluster center, and the data in each cluster is clustered again with k-means. We now have something that looks like a two-level tree of clusters. Of course, this process can be repeated to produce a multi-level tree of clusters. It is easy to use this tree to vector quantize a query data item. We vector quantize at the first level. Doing so chooses a branch of the tree, and we pass the data item to this branch. It is either a leaf, in which case we report the number of the leaf, or it is a set of clusters, in which case we vector quantize, and pass the data item down. This procedure is efficient both when one clusters and at run time.

16.0.4 Using Textons

We can now build a large vocabulary of textons. Details vary from application to application, but the general procedure is clear. We obtain a large number of relevant images (from thousands to millions, depending on the particular application). We extract patches from these images. These patches could have centers that are on a grid, or are randomly chosen, or lie at image corners (which are detected using

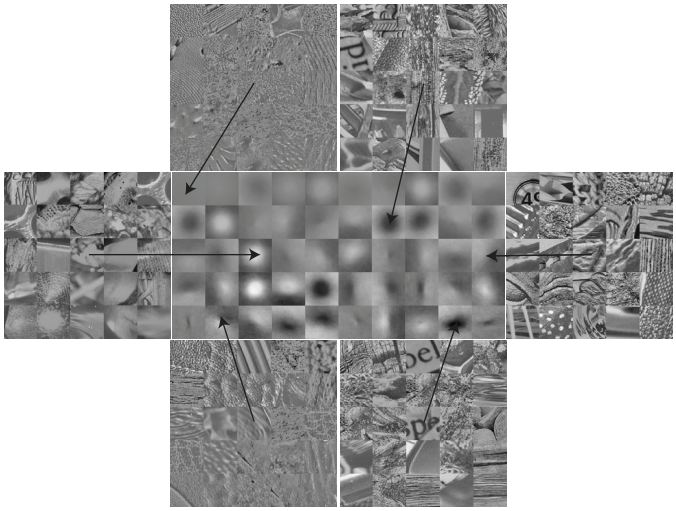

FIGURE 16.4: *Pattern elements can be identified by vector quantizing vectors obtained by reshaping an image window centered on each pixel. Here we show the top 50 pattern elements (or textons), obtained using this strategy from all 1,000 images of the collection of material images described in Figure* **??**. *Each subimage here illustrates a cluster center. For some cluster centers, we show the closest 25 image patches. To measure distance, we first subtracted the average image intensity, and we weighted by a Gaussian to ensure that pixels close to the center of the patch were weighted higher than those far from the center.* This figure shows elements of a database collected by C. Liu, L. Sharan, E. Adelson, and R. Rosenholtz, and published at `http://people.csail.mit.edu/lavanya/research_sharan.html`. Figure by kind permission of the collectors.

methods outside the scope of this discussion). These patches are clustered using k-means or hierarchical k-means. It is quite usual to have k in the hundreds of thousands. We can then describe a new image by decomposing it into patches (typically, on a grid); vector quantizing each patch; and forming a histogram of the centers. Usually, this histogram is normalized, so that larger images (with more patches) do not appear different from smaller images.

When we use this description, we need to keep in mind that it is a histogram. Histograms can be represented as vectors (one entry per bin), but in important respects they are not really vectors. All the entries in a normalized histogram sum to one. This means that measuring the difference between two histograms with a Euclidean distance measure is not a particularly good idea. The significance of a large difference at a particular bin depends on how full the bin is. Write $\mathbf{g}$ and $\mathbf{h}$ for the two histograms we wish to compare, and g_i (resp. h_i) for the i'th bin. Now assume $(h_i - g_i)^2$ is large. If $h_i + g_i$ is large, too, this might not be significant, because we might have a difference in counts that is due to random effects. But if $h_i + g_i$ is small, the difference is much more significant, because one bin has many

entries and another has few. This argues for using a distance

$$d(\mathbf{h}, \mathbf{g}) = \sum_i \frac{(h_i - g_i)^2}{(h_i + g_i)}$$

which is known as the χ^2 distance (often written as the chi-squared distance). The name comes from an analogy with Pearson's χ^2 distance for comparing histograms (which is $\sum_i \frac{(h_i - g_i)^2}{h_i}$).

We can now use this image description in a variety of ways.

- **Image classification with SVM's:** One could use an SVM to classify images. Better results are available with more complex classifier technologies, because SVM's do not account for the histogram distance effect, above.
- **Image classification with nearest neighbors:** ¿A nearest neighbor classifier should use the chi-squared distance. It is often sufficient to use an approximate nearest neighbor method to get a set of possible histograms close to the query, then find the closest using the chi-squared distance.
- **Image retrieval:** There are many applications where we would like to recover images that are similar in appearance to a query image. We can do so by selecting images that have a small chi-squared distance to the query image. Some complex technical paraphenalia are required to make doing so efficient, however.
- **Image clustering:** We can cluster images using the chi-squared distance.

One issue that arises quite regularly in applications is that vector quantization loses information. Small changes in image patch can lead to a large change in the cluster center, and many of these can accumulate to give a big deviation in the histogram. This is an important source of error in applications. There is an effective strategy to handle it. One builds multiple systems, then votes. For example, imagine building a nearest neighbor classifier for images. We build (say) three, rather than one, vector quantizers. Each clusters the same set of image patches, but uses a different random start. We then build three (say) rather than one histogram describing an image, and query three collections, one with each histogram. Finally, we choose the category that gets the majority of votes.

An alternate use of a texton vocabulary is in denoising. Assume we have a large vocabulary of image patches, built as above. A new image comes in, but is noisy, perhaps because there are errors in the storage and retrieval system, or in a communication system, or because the camera is very slightly defocussed. We can build a reconstruction by matching each of the noisy image patches to its cluster center, then replacing the patch with the cluster center. The details are complex, because we need to figure out what to do with patches that overlap, and to ensure that patches that abut one another are consistent.

16.1 EXAMPLES USING VECTOR QUANTIZATION

At `http://archive.ics.uci.edu/ml/datasets/Wholesale+customers`, you will find a dataset giving sums of money spent annually on different commodities by

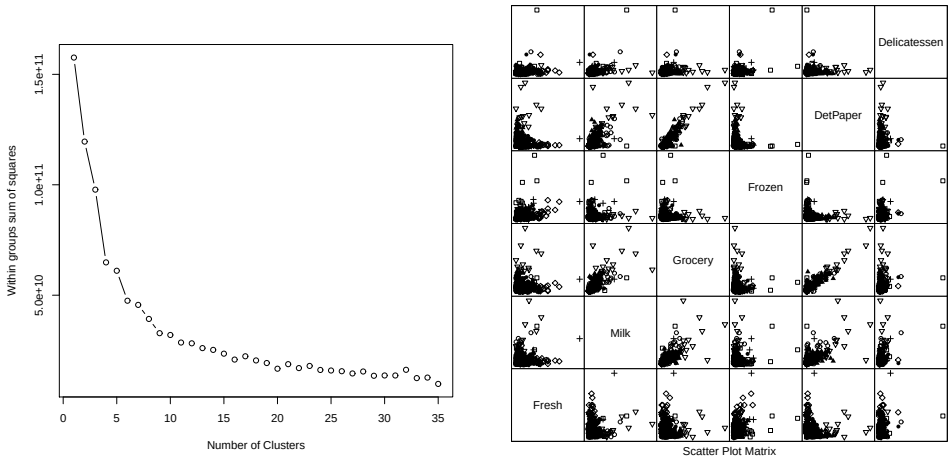

FIGURE 16.5: *On the* **left**, *the sum of squared error for clusterings of the customer data with k-means for k running from 2 to 35. This suggests using a k somewhere in the range 10-30; I chose 20. On the* **right**, *I have clustered this data to 20 cluster centers with k-means. The clusters do seem to be squashed together, but the plot on the left suggests that clusters do capture some important information. Using too few clusters will clearly lead to problems. Notice that I did not scale the data, because each of the measurements is in a comparable unit. For example, wouldn't make sense to scale expenditures on fresh and expenditures on grocery with a different scale.*

customers in Portugal. These customers are divided by channel (two channels) and by region (three regions). You can think of the data as being divided into six groups (one for each channel-region pair). There are 440 records, and so many customers per class. The commodities are divided into a set of categories (fresh; milk; grocery; frozen; detergents and paper; and delicatessen) relevant for the study. Figure 16.6 shows a panel plot of the customer data. Relatively little structure is apparent here — you can't, for example, see any evidence of six clean clusters.

If you think about it, it's too much to expect that all customers in a class are similar. Instead, we expect that there might be different "types" of customer — some might spend more money on milk, others on grocery, and so on. Different groups likely contain different numbers of each type of customer. This would make it hard to see the groups in the panel plot.

Vector quantization can expose this kind of structure rather effectively. I used k-means clustering to cluster the customer data to 20 different clusters (Figure 16.6). Then I described the each group of data by the histogram of customer types that appeared in that group (Figure **??**). Notice how the distinction between the groups is now apparent — the groups do appear to contain quite different distributions of customer type. To be more confident in this analysis, we would need to be sure that different types of customer really are different. We could do this by repeating the analysis for fewer clusters, or by looking at the similarity of customer

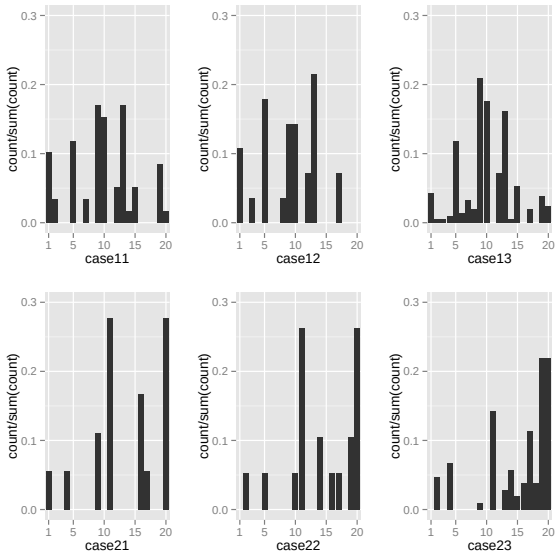

FIGURE 16.6: *The histogram of different types of customer, by group, for the customer data. Notice that different groups do really appear to have different types of customer represented in different amounts.*

types.

Another, more complex, example dataset appears at `https://archive.ics.uci.edu/ml/datasets/Dataset+for+ADL+Recognition+with+Wrist-worn+Accelerometer`. This dataset consists of examples of the signal from a wrist mounted accelerometer, produced as different subjects engaged in different activities of daily life. Activities include: brushing teeth, climbing stairs, combing hair, descending stairs, and so on. Each is performed by sixteen volunteers. The accelerometer samples the data at 32Hz (i.e. this data samples and reports the acceleration 32 times per second). The accelerations are in the x, y and z-directions. Figure 16.7 shows the x-component of various examples of toothbrushing.

There is an important problem with using data like this. Different subjects take quite different amounts of time to perform these activities. For example, some subjects might be more thorough tooth-brushers than other subjects. As another example, people with longer legs walk at somewhat different frequencies than people with shorter legs. This means that the same activity performed by different subjects will produce data vectors *that are of different lengths*. It's not a good idea to deal with this by warping time and resampling the signal. For example, doing so will make a thorough toothbrusher look as though they are moving their hands very fast (or a careless toothbrusher look ludicrously slow: think speeding up or slowing down a movie). So we need a representation that can cope with signals that are a bit longer or shorter than other signals.

Another important property of these signals is that all examples of a particular activity should contain repeated patterns. For example, brushing teeth should show

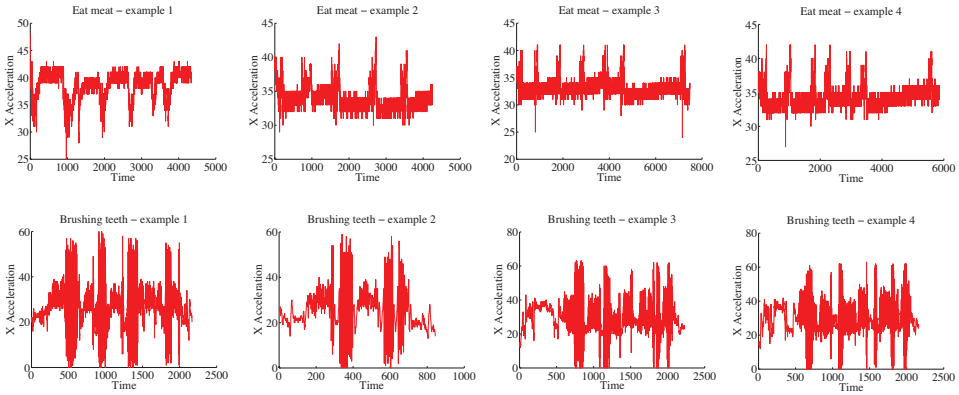

FIGURE 16.7: *Some examples from the accelerometer dataset at* `https://archive.ics.uci.edu/ml/datasets/Dataset+for+ADL+Recognition+with+Wrist-worn+Accelerometer`. *I have labelled each signal by the activity. These show acceleration in the X direction (Y and Z are in the dataset, too). There are four examples for* **brushing teeth** *and four for* **eat meat**. *You should notice that the examples don't have the same length in time (some are slower and some faster eaters, etc.), but that there seem to be characteristic features that are shared within a category (brushing teeth seems to involve faster movements than eating meet).*

fast accelerations up and down; walking should show a strong signal at somewhere around 2 Hz; and so on. These two points should suggest vector quantization to you. Representing the signal in terms of stylized, repeated structures is probably a good idea because the signals probably contain these structures. And if we represent the signal in terms of the relative frequency with which these structures occur, the representation will have a fixed length, even if the signal doesn't. To do so, we need to consider (a) over what time scale we will see these repeated structures and (b) how to ensure we segment the signal into pieces so that we see these structures.

Generally, repetition in activity signals is so obvious that we don't need to be smart about segment boundaries. I broke these signals into 32 sample segments, one following the other. Each segment represents one second of activity. This is long enough for the body to do something interesting, but not so long that our representation will suffer if we put the segment boundaries in the wrong place. This resulted in about 40, 000 segments. I then used hierarchical k-means to cluster these segments. I used two levels, with 40 cluster centers at the first level, and 12 at the second. Figure 16.8 shows some cluster centers at the second level.

I then computed histogram representations for different example signals (Figure 16.9). You should notice that when the activity label is different, the histogram looks different, too.

Another useful way to check this representation is to compare the average within class chi-squared distance with the average between class chi-squared distance. I computed the histogram for each example. Then, for each pair of examples, I computed the chi-squared distance between the pair. Finally, for each pair of *ac-*

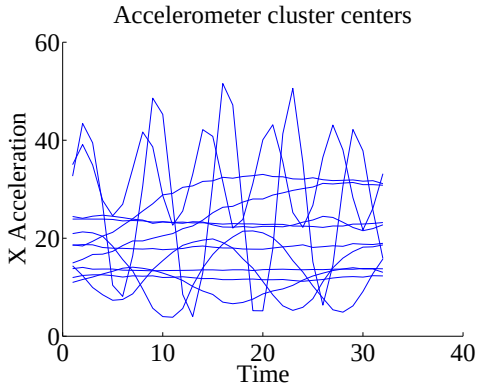

FIGURE 16.8: *Some cluster centers from the accelerometer dataset. Each cluster center represents a one-second burst of activity. There are a total of 480 in my model, which I built using hierarchical k-means. Notice there are a couple of centers that appear to represent movement at about 5Hz; another few that represent movement at about 2Hz; some that look like 0.5Hz movement; and some that seem to represent much lower frequency movement. These cluster centers are samples (rather than chosen to have this property).*

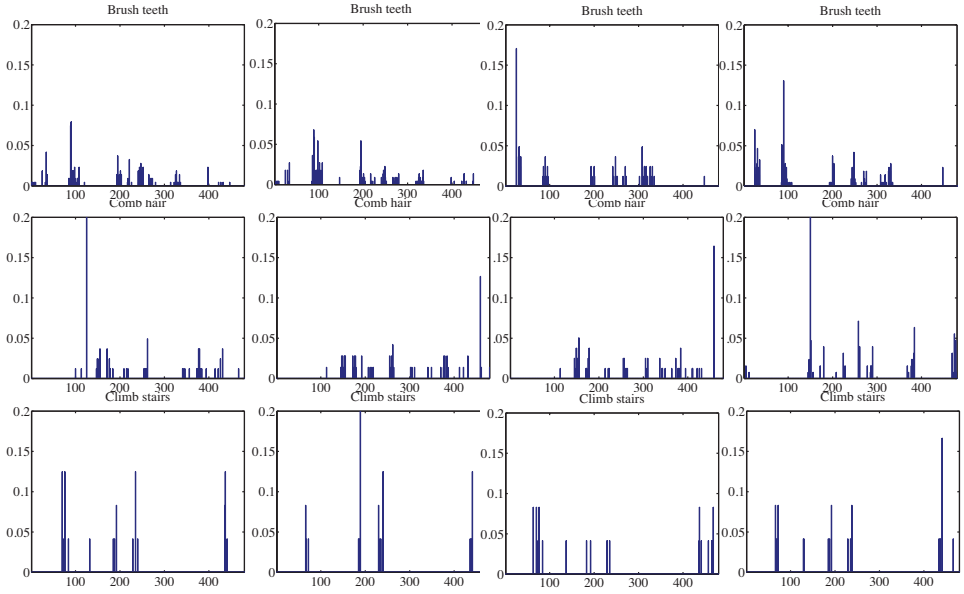

FIGURE 16.9: *Histograms of cluster centers for the accelerometer dataset, for different activities. You should notice that (a) these histograms look somewhat similar for different actors performing the same activity and (b) these histograms look somewhat different for different activities.*

tivity labels, I computed the average distance between pairs of examples where one example has one of the activity labels and the other example has the other activity label. In the ideal case, all the examples with the same label would be very close to one another, and all examples with different labels would be rather different. Table 16.1 shows what happens with the real data. You should notice that for some pairs of activity label, the mean distance between examples is smaller than one would hope for (perhaps some pairs of examples are quite close?). But generally, examples of activities with different labels tend to be further apart than examples of activities with the same label.

0.9	2.0	1.9	2.0	2.0	2.0	1.9	2.0	1.9	1.9	2.0	2.0	2.0	2.0
	1.6	2.0	1.8	2.0	2.0	2.0	1.9	1.9	2.0	1.9	1.9	2.0	1.7
		1.5	2.0	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	2.0
			1.4	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	1.8
				1.5	1.8	1.7	1.9	1.9	1.8	1.9	1.9	1.8	2.0
					0.9	1.7	1.9	1.9	1.8	1.9	1.9	1.9	2.0
						0.3	1.9	1.9	1.5	1.9	1.9	1.9	2.0
							1.8	1.8	1.9	1.9	1.9	1.9	1.9
								1.7	1.9	1.9	1.9	1.9	1.9
									1.6	1.9	1.9	1.9	2.0
										1.8	1.9	1.9	1.9
											1.8	2.0	1.9
												1.5	2.0
													1.5

TABLE 16.1: *Each column of the table represents an activity for the activity dataset* `https://archive.ics.uci.edu/ml/datasets/Dataset+for+ADL+Recognition+with+Wrist-worn+Accelerometer`*, as does each row. In each of the upper diagonal cells, I have placed the average chi-squared distance between histograms of examples from that pair of classes (I dropped the lower diagonal for clarity). Notice that in general the diagonal terms (average within class distance) are rather smaller than the off diagonal terms. This quite strongly suggests we can use these histograms to classify examples successfully.*

Yet another way to check the representation is to try classification with nearest neighbors, using the chi-squared distance to compute distances. I split the dataset into 80 test pairs and 360 training pairs; using 1-nearest neighbors, I was able to get a held-out error rate of 0.79. This suggests that the representation is fairly good at exposing what is important.

16.2 WHAT YOU SHOULD REMEMBER - NEED SOMETHING

CHAPTER 17

Topics and Documents

We often want to deal with free text. For example, we might want to decide which reviewers to trust from a batch of text reviews of restaurants. To do so, we should check for evidence that the reviews are fake. Restaurant owners might pay for positive reviews of their restaurant, or negative reviews of a rivals. These paid reviews are typically quite stylized, because the author gets paid by the review. We might also want to check that the text is consistent with the rating, and that the particular reviewer is active. As another example, we might like to check whether a piece of email is spam or not. In each case, we do not need a particularly deep understanding of what the text *means* as a piece of language — we're just looking for useful superficial evidence to support a decision.

17.1 BASIC TECHNOLOGIES FROM INFORMATION RETRIEVAL

It is quite useful to understand basic ideas in information retrieval. Typical text information retrieval systems expect a set of query words. They use these to query some form of index, producing a list of putative matches. From this list they chose documents with a large enough similarity measure between document and query. These are ranked by a measure of significance, and returned. The important questions to understand here are how to identify which document is similar to the query, and how to score the significance of documents.

17.1.1 Word Counts

Much of text information retrieval is shaped by the fact that a few words are common, but most words are rare. The most common words—typically including "the," "and," "but," "it"—are sometimes called **stop words** and are ignored because almost every document contains many of them. Other words tend to be rare, which means that their frequencies can be quite distinctive. Quite often, it is enough to know whether the word is there or not. For example, documents containing the words "stereo," "fundamental," "trifocal," and "match" are likely to be about 3D reconstruction; documents containing "chrysoprase," "incarnadine," "cinnabarine," and "importunate" are most likely lists of 11 letter words ending in "e" (many such lists exist, for crossword puzzle users; you can check this using Google).

Word Types, Tokens, and Tokenization

Passing from a text document to a set of word counts requires some care. We need to respect the difference between word types — distinct words, however many times they occur — and word tokens — the actual instances of the words. So, for example, the sentence "A cat sat beside another cat" contains five word types and six tokens. For many applications, whether a word type occurs is more important information than the number of times it occurs.

Some nuisance results from the fact that small changes in the form of a word

may not be particularly important. For example, it is unlikely that we need to care about the difference between "cat" and "cats" — we would want to regard these as the same word type. Similarly, we might regard "run" and "running" as the same word type. Quite good programs for reducing words to a standard form (i.e. "laughing" or "laughs" go to "laugh") are known as **stemmers**. Using a stemmer can create problems as well as solve them; for example, a stemmer might regard "stock" and "stocking" as instances of the same word type, thereby confusing avaritia with luxuria in internet searches.

Most words are very rare. One consequence is that, even if we have seen a large collection of documents, we expect to see new word types in new documents. In turn, this means we cannot usually tell in advance all the word types that we need to deal with. This causes a variety of nuisances, particularly when we are dealing with informal text. We must pass from a string of characters to a set of counts for each word type that occurs. But how do we tell whether a string is a word type, a variant word type, or just noise? For example, "youre" is probably the result of poor punctuation; but it takes some background to know that "whilom" is a deliberate archaism, but "wylom" is (at best) a spelling error. Producing a set of word counts from a document is known as **tokenization**. For our purposes, this is the domain of specialized software which we download.

Software pointers: 17.1 *Text tools*

RTextTools is a collection of tools for text data using R, which can be found at `http://www.rtexttools.com`. Notice the variety of examples and tutorials there.
OpenNLP is an extensive set of text processing tools in Java, which can be found at `http://opennlp.apache.org`.
Ingo Feinerer has produced a tokenizer in R, called tm, which can be found at `http://www.inside-r.org/packages/cran/tm/docs/MC_tokenizer`.

Indexing Documents

It is straightforward to build a table representing the documents in which each word occurs, because very few words occur in many documents, so the table is sparse. Write N_w for the number of words and N_d for the number of documents. We could represent the table as an array of lists. There is one list for each word, and the list entries are the documents that contain that word. This object is referred to as an **inverted index**, and can be used to find all documents that contain a logical combination of some set of words. For example, to find all documents that contain any one of a set of words, we would: take each word in the query, look up all documents containing that word in the inverted index, and take the union of the resulting sets of documents. Similarly, we could find documents containing all of the words by taking an intersection, and so on. Such logical queries usually are not sufficient, because the result set might be either very large or too small, and

because we have no notion of which elements of the result set are more important. We need a more refined notion of similarity between documents, and between a document and a query.

Similarity from Word Counts

One measure of similarity for two documents is to compare word frequencies. Assume we have a fixed set of terms that we will work with. We represent each document by a vector $\mathbf{c}$, with one entry for each term. These entries are zero when the term is absent, and contain some measure of the word frequency when the word is present. This measure might be as simple as a one if the word appears at least once in the document, or might be a count of the number of words. Write $\mathbf{c}_1$, $\mathbf{c}_2$ for two such vectors; the **cosine similarity** between the documents they represent is

$$\frac{\mathbf{c}_1 \cdot \mathbf{c}_2}{\|\mathbf{c}_1\| \|\mathbf{c}_2\|}.$$

Two documents that both use an uncommon word are most likely more similar than two documents that both use a common word. We can account for this effect by weighting word counts. The most usual way to do this is called **tf-idf weighting** (for "term frequency-inverse document frequency"). Terms that should have most weight appear often in the particular document we are looking at, but seldom in all documents. Write N_d for the total number of documents and N_t for the number of documents that contain the particular term we are interested in. Then, the inverse document frequency can be estimated as $N_d/(1+N_t)$ (where we add one to avoid dividing by zero). Write $n_t(j)$ for the number of times the term appears in document j and $n_w(j)$ for the total number of words that appear in that document. Then, the tf-idf weight for term t in document j is

$$\left(\frac{n_t(j)}{n_w(j)}\right) / \log\left(\frac{N_d}{(1+N_t)}\right).$$

We divide by the log of the inverse document frequency because we do not want very uncommon words to have excessive weight. Inserting this tf-idf weight into the count vectors above will get a cosine similarity that weights uncommon words that are shared more highly than common words that are shared.

17.1.2 Smoothing Word Counts

Our measurement of similarity will not work well on most real document collections, even if we weight by tf-idf. This is because words tend to be rare, so that most pairs of documents share only quite common words, and so most pairs of documents will have quite small cosine similarity. The real difficulty here is that zero word counts can be underestimates. For example, a document that uses the words "elephant," "tusk," and "pachyderm" should have some affinity for "trunk." If that word does not appear in the document, it is an accident of counting. This means that to measure similarity, we would do well to smooth the word counts.

We can do so by looking at how all terms are distributed across all documents. An alternative representation of the information in an inverted index is as an N_w by N_d table $\mathcal{D}$, where each cell contains an entry if the relevant word is not in the

relevant document and a zero otherwise. Entries could be one if the word occurs, or a count of the number of times the word occurs, or the tf-idf weight for the term in the document. In any case, this table is extremely sparse, so it can be stored and manipulated efficiently. A column of the table is a representation of the words in a document, and the cosine similarity between columns is our original measure of similarity between documents.

Zeros in $\mathcal{D}$ might be the result of counting accidents, as above. We would like a version of this table that smooths word counts. We assume that, if two documents are on the same topic, then they tend to use the same words in about the same frequency. This means that in the smoothed version of the table, there are many columns that are similar. Now think about a word that is associated with a topic (say, "elephant"). Then in the smoothed version of the table, it will appear in documents about elephants, and not in documents about something else. But this will be true of other such words (say, "ivory"). This means that, in the smoothed table, many rows should be similar to one another as well.

The smoothed version of $\mathcal{D}$ will be significantly rank-deficient because many columns (resp. rows) are similar. We can estimate the smoothed version by computing a singular value decomposition of $\mathcal{D}$ as $\mathcal{D} = \mathcal{U}\Sigma\mathcal{V}^T$. Write $\mathcal{U}_k$ for the matrix consisting of the first k columns of $\mathcal{U}$, $\mathcal{V}_k$ for the matrix consisting of the first k columns of $\mathcal{V}$, Σ_k for Σ with all but the k largest singular values set to be zero, and write $\hat{\mathcal{D}} = \mathcal{U}_k\Sigma_k\mathcal{V}_k^T$.

Now consider the ith column of $\mathcal{D}$, which we write as $\mathbf{d}_i$. The corresponding column $\hat{\mathbf{d}}_i$ of $\hat{\mathcal{D}}$ lies in the span of $\mathcal{U}_k$. The word counts are smoothed by forcing them to lie in this span. For example, assume that there are many documents discussing elephants, and only one uses the word "pachyderm." The count vectors for each of these documents could be represented by a single column, but error will be minimized if there is a small count for "pachyderm" in each. Because of this smoothing effect, cosine distances between documents represented by columns of $\hat{\mathcal{D}}$ are a much more reliable guide to similarity.

To compute cosine similarity between an old document and a new document with count vector $\mathbf{q}$, we project the new document's count vector onto the columns of $\mathcal{U}_k$ to obtain $\hat{\mathbf{q}} = \mathcal{U}_k\mathcal{U}_k^T\mathbf{q}$. We can then take the inner product of $\hat{\mathbf{q}}$ and $\hat{\mathbf{d}}_i$. A complete table of inner products (cosine distances) between documents is given by

$$\hat{\mathcal{D}}^T\hat{\mathcal{D}} = (\mathcal{V}_k\Sigma_k)(\Sigma_k\mathcal{V}^T) = (\Sigma_k\mathcal{V}_k^T)^T(\Sigma_k\mathcal{V}^T),$$

so that we can think of the columns of $\Sigma_k\mathcal{V}^T$ as points in a k-dimensional "concept space" that represents the inner products exactly. One could, for example, cluster documents in this space rather than the original count space, and expect a better clustering. Computing the SVD of $\mathcal{D}$ is known as **latent semantic analysis**; using the concept space for indexing is known as **latent semantic indexing**.

$\hat{\mathcal{D}}$ is useful in other ways, too. There is a rough tendency of words that have similar meaning to appear near similar words in similar documents, an idea known as **distributional semantics**. This means that cosine similarity between *rows* of $\hat{\mathcal{D}}$ is an estimate of the similarity of meaning of two terms, because it counts the extent to which they co-occur. Furthermore, $\hat{\mathcal{D}}$ can be used as an inverted index. If we use it in this way, we are not guaranteed that every document recovered contains all the words we used in the query; instead, it might contain very similar words.

This is usually a good thing. The columns of $\mathcal{U}$ are sometimes called **topics** and can be thought of as model word frequency vectors; the coordinates of a column in the semantic space show the weights with which topics should be mixed to obtain the document.

17.2 TOPIC MODELS

In the previous section, we assumed that documents that were similar had similar smoothed word counts. We estimated a word-document matrix using this idea. This allowed us to measure the cosine similarity between two documents more reliably. We can expand this idea into a **topic model** – a model of how words that appear in a document result from the topic of the document.

17.2.1 A Multinomial Topic Model

Topic models are built to represent how a document is generated. Our first assumption is that the order in which words appear does not matter. The only evidence we will use is the number of times each word appears. This assumption clearly doesn't describe language very well, but it seems to be a quite reliable guide to what documents are about. You should notice that we used it in our model of information retrieval, too. The assumption is known as the **bag-of-words** model.

Now assume we wish to produce a document. We will assume that each document treats only one topic. We assume that there are k possible topics. We produce a document by selecting a topic, then producing words from that topic. We assume that the topic is chosen by sampling from a multinomial distribution, so we draw topic t with probability $P(t)$.

Now the topic is known. We assume that the counts for each word are independent of the counts for other words, and depend only on the topic. These assumptions make estimating the parameters of the model much easier. They're also a tolerable model of how words might be used in producing a document. For example, if you are writing about "elephants", the rate at which you use (or forget to use) the word "pachyderm" is likely to be independent from the rate at which you use (or forget to use) the word "tusk". And the rate at which you use the word "pachyderm" is higher when you write about "elephants" or "Victorian zookeepers", but much lower when you write about anything else.

This assumption means that, when the topic has been selected, the words are obtained by sampling from a multinomial distribution. We can write $P(w|t)$ for the probability of getting a particular word w conditioned on a topic t. Notice that there is one such distribution per topic.

In the usual case, we have a collection of documents and need to determine $P(t)$ and $P(w|t)$ for each topic and word. From this information we can build a representation of each document. A rigorous description of the procedure for inferring $P(t)$ and $P(w|t)$ is beyond the scope of this description, because it requires some slightly finicky probabilistic reasoning, and because in practice we use more complex models anyhow. But it's quite easy to get a good sense of how the method works, which has a strong analogy to k-means.

Assume for the moment we know which topic each document belongs to. Write N_d for the number of documents in the collection. Then it is easy to estimate $P(t)$

by counting. For each topic t, we have the estimate

$$P(t) = \frac{\#\text{ documents with topic } t}{N_d}.$$

Similarly, it is easy to estimate $P(w|t)$ by counting. For each word type w, we know how many times it occurs in a document of topic t. Write $N_w(t)$ for the total number of word tokens that are in documents of type t. So we can estimate

$$P(w|t) = \frac{\#\text{ of times } w \text{ occurs in documents with topic } t}{N_w(t)}.$$

Now let us work the other way round. Assume we know $P(t)$ and $P(w|t)$. Then for each document, we can determine the posterior probability that document came from topic t. Write $\mathbf{w}(d)$ for a vector of word counts (ie for each word in the vocabulary, there is one component in the vector; this contains the number of times that word appears in the document d), and w_i for the i'th component of that vector. We want to know

$$\begin{aligned} P(\text{topic} = t|\mathbf{w}(d)) &= \frac{P(\mathbf{w}(d)|t)P(t)}{P(\mathbf{w}(d))} \\ &= \frac{\prod_i P(w_i|t)P(t)}{\sum_u \prod_i P(w_i|t=u)P(t=u)} \end{aligned}$$

which we can evaluate. We could assign the document to the topic with highest posterior. In fact, we can do better. We could regard this posterior distribution as a "soft count", assigning the document to topic t with weight $P(t|\mathbf{w}(d))$. We can estimate the distributions $P(t)$ and $P(w|t)$ by regarding these soft counts as fractional counts (compare section **??**).

So we have an algorithm, very like k-means. We make an initial guess of $P(w|t)$, typically by choosing one document at random to estimate the topic centers, then counting the words in the document. We can then get an initial guess at $P(t)$ as a uniform distribution. We then form soft assignments of each document to each topic, and use these to re-estimate the probabilities. We iterate this procedure until it converges. With a little care, it will converge. There are two things to be careful about. We may be unlucky, and have a topic that gets no documents at a particular iteration. This will make it hard to estimate $P(w|t)$. Just as in k means, we could resolve this by choosing a document at random and assigning it to that topic. We may also find that in a particular topic, a word appears 0 times. This is likely a miscount. In reality, the word is uncommon given the topic, and is so uncommon that it has not appeared. But it is highly unlikely that it *cannot* appear, which is what setting $P(w|t) = 0$ would imply. There are a variety of sophisticated tricks we could apply to this situation. The simplest, which is often quite successful, is to add one to every count.

Now assume we have estimated $P(t)$ and all the $P(w|t)$. We can use this information to describe any document in a variety of ways. We could simply report the most likely topic given the document. We could report $P(w|t)$ for that topic. A more detailed description is given by computing $P(\text{topic} = t|\mathbf{w}(d))$. You can think of this as a vector of affinities connecting the document to each topic. For example,

a document might deal with "elephants" with weight 0.2 and "rhinoceroses" with weight 0.02.

17.2.2 Latent Dirichlet Allocation

The difficulty with the approach above is that it assumes that each document has one topic. We can measure the affinity between a document and multiple topics, but the assumptions rule out documents that (say) each discuss two distinct topics ("elephants"' and then "marula", say). It would be better to allow each document to deal with multiple topics.

We can do so with the following model. Assume we wish to generate a collection of documents. To do so, we will first generate a set of topics. We represent each with a vector $P(w|t)$. We obtain these vectors by sampling from a prior distribution on topic vectors. Once we have these vectors, we must obtain words for each document. The document first chooses a set of topic weights. This is the probability that a word will be chosen from a topic *for that document* — so each document may have a different set of topic weights. For example, one document may use words from the "elephants" topic more often and another might use words from the "rhinoceroses" topic more often. The set of topic weights is a vector of probabilities, one per topic, and each document has one.

Once the document has a set of topic weights, it must generate words. It does so by repeatedly: choosing a topic at random using the vector of topic weights, then visiting that topic and choosing a single word at random using that topic's $P(w|t)$. As a result, words in a document can come from different topics. In fact, two instances of the same word may come from two different topics.

Assume we have a set of documents that we believe were generated using this model. We must determine the model parameters. These are the word probabilities for each topic, and the topic weights for each document. How we fit this model to a set of documents is way beyond the scope of this text, but the general picture is rather like multinomial topic models. If we knew which topic every word came from, we could recover the word probabilities and the topic weights easily by counting. Similarly, if we knew what the topic weights and the word probabilities were, we could estimate which topic each word comes from. This is a fairly substantial problem in large scale inference, because we might be looking at millions of words and thousands of topics. A variety of good software is available freely on the web (Section **??**).

For our purposes, these models are important because they give us sophisticated representations of what documents might be about. Consider the document topic weights. Documents dealing in different ways with different matters are likely to have different topic weights. With a modicum of luck, documents treating the same topic using different approaches ("elephant conservation" vs. "elephant cookery", say) will have different topic weights, too. In turn, this means we have a good chance of drawing useful conclusions about documents using these topic weights. For example, they might be used to classify or cluster documents.

Software pointers: 17.2 *Latent Dirichlet Allocation software*

David Blei's research group has produced a great deal of topic modelling software. A guide can be found at `https://www.cs.princeton.edu/~blei/topicmodeling.html`. There is software in C, C++, Python and R.
Mark Steyvers and Tom Griffiths have produced a topic modelling toolbox for Matlab. The code, and a series of example scripts for a variety of topic models, can be found at `http://psiexp.ss.uci.edu/research/programs_data/toolbox.htm`.
Bettina Grün and Kurt Hornik have produced an interface to Blei's C/C++ code for R, called topicmodels. Details can be found at `http://cran.r-project.org/web/packages/topicmodels/index.html`.
There is a nice tutorial on using this package together with RTextTools at `http://www.rtexttools.com/1/post/2011/08/getting-started-with-latent-dirichlet-allocation-using-rtexttools-topicmodels.html`.

CHAPTER 18

Math Resources

18.1 USEFUL MATERIAL ABOUT MATRICES

Terminology:

- A matrix $\mathcal{M}$ is **symmetric** if $\mathcal{M} = \mathcal{M}^T$. A symmetric matrix is necessarily square.
- We write $\mathcal{I}$ for the identity matrix.
- A matrix is **diagonal** if the only non-zero elements appear on the diagonal. A diagonal matrix is necessarily symmetric.
- A symmetric matrix is **positive semidefinite** if, for any $\mathbf{x}$ such that $\mathbf{x}^T\mathbf{x} > 0$ (i.e. this vector has at least one non-zero component), we have $\mathbf{x}^T\mathcal{M}\mathbf{x} \geq 0$.
- A symmetric matrix is **positive definite** if, for any $\mathbf{x}$ such that $\mathbf{x}^T\mathbf{x} > 0$, we have $\mathbf{x}^T\mathcal{M}\mathbf{x} > 0$.
- A matrix $\mathcal{R}$ is **orthonormal** if $\mathcal{R}^T\mathcal{R} = \mathcal{I} = \mathcal{I}^T = \mathcal{R}\mathcal{R}^T$. Orthonormal matrices are necessarily square.

Orthonormal matrices: You should think of orthonormal matrices as rotations, because they do not change lengths or angles. For $\mathbf{x}$ a vector, $\mathcal{R}$ an orthonormal matrix, and $\mathbf{u} = \mathcal{R}\mathbf{x}$, we have $\mathbf{u}^T\mathbf{u} = \mathbf{x}^T\mathcal{R}^T\mathcal{R}\mathbf{x} = \mathbf{x}^T\mathcal{I}\mathbf{x} = \mathbf{x}^T\mathbf{x}$. This means that $\mathcal{R}$ doesn't change lengths. For $\mathbf{y}$, $\mathbf{z}$ both unit vectors, we have that the cosine of the angle between them is $\mathbf{y}^T\mathbf{x}$; but, by the same argument as above, the inner product of $\mathcal{R}\mathbf{y}$ and $\mathcal{R}\mathbf{x}$ is the same as $\mathbf{y}^T\mathbf{x}$. This means that $\mathcal{R}$ doesn't change angles, either.

Eigenvectors and Eigenvalues: Assume $\mathcal{S}$ is a $d \times d$ symmetric matrix, $\mathbf{v}$ is a $d \times 1$ vector, and λ is a scalar. If we have

$$\mathcal{S}\mathbf{v} = \lambda\mathbf{v}$$

then $\mathbf{v}$ is referred to as an **eigenvector** of $\mathcal{S}$ and λ is the corresponding **eigenvalue**. Matrices don't have to be symmetric to have eigenvectors and eigenvalues, but the symmetric case is the only one of interest to us.

In the case of a symmetric matrix, the eigenvalues are real numbers, and there are d distinct eigenvectors that are normal to one another, and can be scaled to have unit length. They can be stacked into a matrix $\mathcal{U} = [\mathbf{v}_1, \ldots, \mathbf{v}_d]$. This matrix is orthonormal, meaning that $\mathcal{U}^T\mathcal{U} = \mathcal{I}$. This means that there is a diagonal matrix Λ such that

$$\mathcal{S}\mathcal{U} = \mathcal{U}\Lambda.$$

In fact, there is a large number of such matrices, because we can reorder the eigenvectors in the matrix $\mathcal{U}$, and the equation still holds with a new Λ, obtained by reordering the diagonal elements of the original Λ. There is no reason to keep track of this complexity. Instead, we adopt the convention that the elements of $\mathcal{U}$ are always ordered so that the elements of Λ are sorted along the diagonal, with the largest value coming first.

Diagonalizing a symmetric matrix: This gives us a particularly important procedure. We can convert any symmetric matrix $\mathcal{S}$ to a diagonal form by computing

$$\mathcal{U}^T \mathcal{S} \mathcal{U} = \Lambda.$$

This procedure is referred to as **diagonalizing** a matrix. Again, we assume that the elements of $\mathcal{U}$ are always ordered so that the elements of Λ are sorted along the diagonal, with the largest value coming first. Diagonalization allows us to show that positive definiteness is equivalent to having all positive eigenvalues, and positive semidefiniteness is equivalent to having all non-negative eigenvalues.

Factoring a matrix: Assume that $\mathcal{S}$ is symmetric and positive semidefinite. We have that

$$\mathcal{S} = \mathcal{U} \Lambda \mathcal{U}^T$$

and all the diagonal elements of Λ are non-negative. Now construct a diagonal matrix whose diagonal entries are the positive square roots of the diagonal elements of Λ; call this matrix $\Lambda^{(1/2)}$. We have $\Lambda^{(1/2)}\Lambda^{(1/2)} = \Lambda$ and $(\Lambda^{(1/2)})^T = \Lambda^{(1/2)}$. Then we have that

$$\mathcal{S} = (\mathcal{U}\Lambda^{(1/2)})(\Lambda^{(1/2)}\mathcal{U}^T) = (\mathcal{U}\Lambda^{(1/2)})(\mathcal{U}\Lambda^{(1/2)})^T$$

so we can factor $\mathcal{S}$ into the form $\mathcal{X}\mathcal{X}^T$ by computing the eigenvectors and eigenvalues.

18.1.1 Approximating A Symmetric Matrix

Assume we have a $k \times k$ symmetric matrix $\mathcal{T}$, and we wish to construct a matrix $\mathcal{A}$ that approximates it. We require that (a) the rank of $\mathcal{A}$ is precisely $r < k$ and (b) the approximation should minimize the **Frobenius norm**, that is,

$$\| (\mathcal{T} - \mathcal{A}) \|_F^2 = \sum_{ij} (T_{ij} - A_{ij})^2.$$

It turns out that there is a straightforward construction that yields $\mathcal{A}$.

The first step is to notice that if $\mathcal{U}$ is orthonormal and $\mathcal{M}$ is any matrix, then

$$\| \mathcal{U}\mathcal{M} \|_F = \| \mathcal{M}\mathcal{U} \|_F = \| \mathcal{M} \|_F.$$

This is true because $\mathcal{U}$ is a rotation (as is $\mathcal{U}^T = \mathcal{U}^{-1}$), and rotations do not change the length of vectors. So, for example, if we write $\mathcal{M}$ as a table of row vectors $\mathcal{M} = [\mathbf{m}_1, \mathbf{m}_2, ...\mathbf{m}_k]$, then $\mathcal{U}\mathcal{M} = [\mathcal{U}\mathbf{m}_1, \mathcal{U}\mathbf{m}_2, ...\mathcal{U}\mathbf{m}_k]$. Now $\| \mathcal{M} \|_F^2 = \sum_{j=1}^k \| \mathbf{m}_j \|^2$, so $\| \mathcal{U}\mathcal{M} \|_F^2 = \sum_{i=1}^k \| \mathcal{U}\mathbf{m}_k \|^2$. But rotations do not change lengths, so $\| \mathcal{U}\mathbf{m}_k \|^2 = \| \mathbf{m}_k \|^2$, and so $\| \mathcal{U}\mathcal{M} \|_F = \| \mathcal{M} \|_F$. To see the result for the case of $\mathcal{M}\mathcal{U}$, just think of $\mathcal{M}$ as a table of row vectors.

Notice that, if $\mathcal{U}$ is the orthonormal matrix whose columns are eigenvectors of $\mathcal{T}$, then we have

$$\|(\mathcal{T}-\mathcal{A})\|_F{}^2 = \|\mathcal{U}^T(\mathcal{T}-\mathcal{A})\mathcal{U}\|_F{}^2.$$

Now write Λ_r for $\mathcal{U}^T\mathcal{A}\mathcal{U}$, and Λ for the diagonal matrix of eigenvalues of $\mathcal{T}$. Then we have

$$\|(\mathcal{T}-\mathcal{A})\|_F{}^2 = \|\Lambda-\Lambda_A\|_F{}^2,$$

an expression that is easy to solve for Λ_A. We know that Λ is diagonal, so the best Λ_A is diagonal, too. The rank of $\mathcal{A}$ must be r, so the rank of Λ_A must be r as well. To get the best Λ_A, we keep the r largest diagonal values of Λ, and set the rest to zero; Λ_A has rank r because it has only r non-zero entries on the diagonal, and every other entry is zero.

Now to recover $\mathcal{A}$ from Λ_A, we know that $\mathcal{U}^T\mathcal{U} = \mathcal{U}\mathcal{U}^T = \mathcal{I}$ (remember, $\mathcal{I}$ is the identity). We have $\Lambda_A = \mathcal{U}^T\mathcal{A}\mathcal{U}$, so

$$\mathcal{A} = \mathcal{U}\Lambda_A\mathcal{U}^T.$$

We can clean up this representation in a useful way. Notice that only the first r columns of $\mathcal{U}$ (and the corresponding rows of $\mathcal{U}^T$) contribute to $\mathcal{A}$. The remaining $k-r$ are each multiplied by one of the zeros on the diagonal of Λ_A. Remember that, by convention, Λ was sorted so that the diagonal values are in descending order (i.e. the largest value is in the top left corner). We now keep only the top left $r \times r$ block of Λ_A, which we write Λ_r. We then write $\mathcal{U}_r$ for the $k \times r$ matrix consisting of the first r columns of $\mathcal{U}$. Then

$$\mathcal{A} = \mathcal{U}_r\Lambda_r\mathcal{U}^T$$

This is so useful a result, I have displayed it in a box; you should remember it.

Procedure: 18.1 *Approximating a symmetric matrix with a low rank matrix*

Assume we have a symmetric $k \times k$ matrix $\mathcal{T}$. We wish to approximate $\mathcal{T}$ with a matrix $\mathcal{A}$ that has rank $r < k$. Write $\mathcal{U}$ for the matrix whose columns are eigenvectors of $\mathcal{T}$, and Λ for the diagonal matrix of eigenvalues of $\mathcal{A}$ (so $\mathcal{A}\mathcal{U} = \mathcal{U}\Lambda$). Remember that, by convention, Λ was sorted so that the diagonal values are in descending order (i.e. the largest value is in the top left corner).
Now construct Λ_r from Λ by setting the $k-r$ smallest values of Λ to zero, and keeping only the top left $r \times r$ block. Construct $\mathcal{U}_r$, the $k \times r$ matrix consisting of the first r columns of $\mathcal{U}$. Then

$$\mathcal{A} = \mathcal{U}_r\Lambda_r\mathcal{U}_r^T$$

is the best possible rank r approximation to $\mathcal{T}$ in the Frobenius norm.

Now if $\mathcal{A}$ is positive semidefinite (i.e. if at least the r largest eigenvalues of $\mathcal{T}$ are non-negative), then we can factor $\mathcal{A}$ as in the previous section. This yields a procedure to approximate a symmetric matrix by factors. This is so useful a result, I have displayed it in a box; you should remember it.

Procedure: 18.2 *Approximating a symmetric matrix with low dimensional factors*

Assume we have a symmetric $k \times k$ matrix $\mathcal{T}$. We wish to approximate $\mathcal{T}$ with a matrix $\mathcal{A}$ that has rank $r < k$. We assume that at least the r largest eigenvalues of $\mathcal{T}$ are non-negative. Write $\mathcal{U}$ for the matrix whose columns are eigenvectors of $\mathcal{T}$, and Λ for the diagonal matrix of eigenvalues of $\mathcal{A}$ (so $\mathcal{A}\mathcal{U} = \mathcal{U}\Lambda$). Remember that, by convention, Λ was sorted so that the diagonal values are in descending order (i.e. the largest value is in the top left corner).
Now construct Λ_r from Λ by setting the $k - r$ smallest values of Λ to zero and keeping only the top left $r \times r$ block. Construct $\Lambda_r^{(1/2)}$ by replacing each diagonal element of Λ with its positive square root. Construct $\mathcal{U}_r$, the $k \times r$ matrix consisting of the first r columns of $\mathcal{U}$. Then write $\mathcal{V} = (\mathcal{U}_r \Lambda_r^{(1/2)})$

$$\mathcal{A} = \mathcal{V}\mathcal{V}^T$$

is the best possible rank r approximation to $\mathcal{T}$ in the Frobenius norm.