Chapter 5

Pause to Reflect

Now that we have some basics, let's step back a bit, and think about what we've learned.

5.1 A Cautionary Tale

Intuition is great, but it can lead you astray! The example in this section shows how things can go wrong.

5.1.1 Trick Coins, Tricky Example

Suppose we have two trick coins in a box. They look identical, but one of them, denoted coin 1, is heavily weighted toward heads, with a 0.9 probability of heads, while the other, denoted coin 2, is biased in the opposite direction, with a 0.9 probability of tails. Let C_1 and C_2 denote the events that we get coin 1 or coin 2, respectively.

Our experiment consists of choosing a coin at random from the box, and then tossing it n times. Let B_i denote the outcome of the i^{th} toss, i = 1,2,3,..., where $B_i = 1$ means heads and $B_i = 0$ means tails. Let $X_i = B_1 + ... + B_i$, so X_i is a count of the number of heads obtained through the i^{th} toss.

The question is: "Does the random variable X_i have a binomial distribution?" Or, more simply, the question is, "Are the random variables B_i independent?" To most people's surprise, the answer is No (to both questions). Why not?

The variables B_i are indeed 0-1 variables, and they have a common success probability. But they are not independent! Let's see why they aren't.

Consider the events $A_i = \{B_i = 1\}$, i = 1,2,3,... In fact, just look at the first two. By definition, they are independent if and only if

$$P(A_1 \text{ and } A_2) = P(A_1)P(A_2) \tag{5.1}$$

First, what is $P(A_1)$? **Now, wait a minute!** Don't answer, "Well, it depends on which coin we get," because this is NOT a conditional probability. Yes, the *conditional* probabilities $P(A_1|C_1)$ and $P(A_1|C_2)$ are 0.9 and 0.1, respectively, but the *unconditional* probability is $P(A_1) = 0.5$. You can deduce that either by the symmetry of the situation, or by

$$P(A_1) = P(C_1)P(A_1|C_1) + P(C_2)P(A_1|C_2) = (0.5)(0.9) + (0.5)(0.1) = 0.5 \tag{5.2}$$

You should think of all this in the notebook context. Each line of the notebook would consist of a report of three things: which coin we get; the outcome of the first toss; and the outcome of the second toss. (Note by the way that in our experiment we don't know which coin we get, but conceptually it should have a column in the notebook.) If we do this experiment for many, many lines in the notebook, about 90% of the lines in which the coin column says "1" will show Heads in the second column. But 50% of the lines *overall* will show Heads in that column.

So, the right hand side of Equation (5.1) is equal to 0.25. What about the left hand side?

$$\begin{aligned} P(A_1 \text{ and } A_2) &= P(A_1 \text{ and } A_2 \text{ and } C_1) + P(A_1 \text{ and } A_2 \text{ and } C_2) && (5.3)\\ &= P(A_1 \text{ and } A_2|C_1)P(C_1) + P(A_1 \text{ and} A_2|C_2)P(C_2) && (5.4)\\ &= (0.9)^2(0.5) + (0.1)^2(0.5) && (5.5)\\ &= 0.41 && (5.6) \end{aligned}$$

Well, 0.41 is not equal to 0.25, so you can see that the events are not independent, contrary to our first intuition. And that also means that X_i is not binomial.

5.1.2 Intuition in Retrospect

To get some intuition here, think about what would happen if we tossed the chosen coin 10000 times instead of just twice. If the tosses were independent, then for example knowledge of the first 9999 tosses should not tell us anything about the 10000th toss. But that is not the case at all. After 9999 tosses, we are going to have a very good idea as to which coin we had chosen, because by that time we will have gotten about 9000 heads (in the case of coin C_1) or about 1000 heads (in the case of C_2). In the former case, we know that the 10000th toss is likely to be a head, while

in the latter case it is likely to be tails. **In other words, earlier tosses do indeed give us information about later tosses, so the tosses aren't independent.**

5.1.3 Implications for Modeling

The lesson to be learned is that independence can definitely be a tricky thing, not to be assumed cavalierly. And in creating probability models of real systems, we must give very, very careful thought to the conditional and unconditional aspects of our models—-it can make a huge difference, as we saw above. Also, the conditional aspects often play a key role in formulating models of nonindependence.

This trick coin example is just that—tricky—but similar situations occur often in real life. If in some medical study, say, we sample people at random from the population, the people are independent of each other. But if we sample *families* from the population, and then look at children within the families, the children within a family are not independent of each other.

5.2 What About "Iterated Variance"?

In analogy to (4.68), one would think that

$$Var(V) = \sum_c P(U = c)\ Var(V \mid U = c) \tag{5.7}$$

In terms of the student heights example above, this wouldn't make sense, because it doesn't take into account the variation in means from one department to another. (This matter, because $Var(V) = E[(V - EV)^2]$.) So, it's not surprising that another term must be added to (5.7). This topic is discussed in Section **??**, but for now the point is that although good intuition is essential, we must not overrely on it..

5.3 Why Not Just Do All Analysis by Simulation?

Now that computer speeds are so fast, one might ask why we need to do mathematical probability analysis; why not just do everything by simulation? There are a number of reasons:

- Even with a fast computer, simulations of complex systems can take days, weeks or even months.
- Mathematical analysis can provide us with insights that may not be clear in simulation.

Chapter 6

Introduction to Discrete Markov Chains

(From this point onward, we will be making heavy use of linear algebra. The reader may find it helpful to review Appendix B.)

Here we introduce Markov chains, a topic covered in much more detail in Chapter 12.

The basic idea is that we have random variables $X_1, X_2, ...$, with the index representing time. Each one can take on any value in a given set, called the **state space**; X_n is then the **state** of the system at time n. The state space is assumed either finite or **countably infinite**.[1]

We sometimes also consider an initial state, X_0, which might be modeled as either fixed or random. However, this seldom comes into play.

The key assumption is the **Markov property**, which in rough terms can be described as:

> The probabilities of future states, given the present state and the past state, depends only on the present state; the past is irrelevant.

In formal terms:

$$P(X_{t+1} = s_{t+1} | X_t = s_t, X_{t-1} = s_{t-1}, \ldots, X_0 = s_0) = P(X_{t+1} = s_{t+1} | X_t = s_t) \tag{6.1}$$

[1]The latter is a mathematical term meaning, in essence, that it is possible to denote the space using integer subscripts. It can be shown that the set of all real numbers is not countably infinite, though perhaps surprisingly the set of all rational numbers *is* countably infinite.

Note that in (6.1), the two sides of the equation are equal but their common value may depend on t. We assume that this is not the case; nondependence on t is known as **stationarity**.[2] For instance, the probability of going from state 2 to state 5 at time 29 is assumed to be the same as the corresponding probability at time 333.

6.1 Matrix Formulation

We define p_{ij} to be the probability of going from state i to state j in one time step; note that this is a *conditional* probability, i.e. $P(X_{n+1} = j \mid X_n = i)$. These quantities form a matrix P,[3] whose row i, column j element is p_{ij}, which is called the **transition matrix**. Each row of P must sum to 1 (do you see why?).

For example, consider a three-state Markov chain with transition matrix

$$P = \begin{pmatrix} \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ 1 & 0 & 0 \end{pmatrix} \tag{6.2}$$

This means that if we are now at state 1, the probabilities of going to states 1, 2 and 3 are 1/2, 0 and 1/2, respectively. Note that each row's probabilities must sum to 1—after all, from any particular state, we must go *somewhere*.

Actually, the m^{th} power, P^m, of the transition matrix gives the probabilities for m-step transitions. In other words, the (i,j) element of P^m is $P(X_{t+m} = j|X_t = i)$. This is clear for the case m = 2 (after which one can use mathematical induction), as follows.

As usual, "break big events down into small events." How can it happen that $X_{t+2} = j$? Well, break things down according to where we might go first after leaving i. We might go from i to 1, say, then from 1 to j. So,

$$P(X_{t+2} = j|X_t = i) = \sum_k p_{ik}\ p_{kj} \tag{6.3}$$

In view of the rule for multiplying matrices, the expression on the right-hand side is simply the (i,j) element of P^2!

[2] Not to be confused with the notion of a stationary distribution, coming below.

[3] Unfortunately, we have some overloading of symbols here. Both in this book and in the field in general, we usually use the letter P to denote this matrix, yet we continue to denote probabilities by P( ). However, it will be clear from context which we mean. The same is true for our transition probabilities p_{ij}, which use a subscripted letter p, which is also the case for probably mass functions.

6.2 Example: Die Game

As our first example of Markov chains, consider the following game. One repeatedly rolls a die, keeping a running total. Each time the total exceeds 10, we receive one dollar, and continue playing, resuming where we left off, mod 10. Say for instance we have a total of 8, then roll a 5. We receive a dollar, and now our total is 3.

It will simplify things if we assume that the player starts with one free point.

This process clearly satisfies the Markov property, with our state being our current total. If our current total is 6, for instance, then the probability that we next have a total of 9 is 1/6, *regardless of what happened our previous rolls.* We have p_{25}, p_{72} and so on all equal to 1/6, while for instance $p_{29} = 0$. Here's the code to find the transition matrix P:

```
1  # 10 states, so 10x10 matrix
2  # since most elements will be 0s, set them all to 0 first,
3  # then replace those that should be nonzero
4  p <- matrix(rep(0,100),nrow=10)
5  onesixth <- 1/6
6  for (i in 1:10) {  # look at each row
7     # since we are rolling a die, there are only 6 possible states we
8     # can go to from i, so check these
9     for (j in 1:6) {
10       k <- i + j  # new total, but did we win?
11       if (k > 10) k <- k - 10
12       p[i,k] <- onesixth
13    }
14 }
```

Note that since we knew that many entries in the matrix would be zero, it was easier just to make them all 0 first, and then fill in the nonzero ones.

6.3 Long-Run State Probabilities

Let N_{it} denote the number of times we have visited state i during times 1,...,t. For instance, in the die game, $N_{8,22}$ would be the number of rolls among the first 22 that resulted in our having a total of 8.

In typical applications we have that

$$\pi_i = \lim_{t\to\infty} \frac{N_{it}}{t} \tag{6.4}$$

exists for each state i. Under a couple more conditions,[4] we have the stronger result,

$$\lim_{t\to\infty} P(X_t = i) = \pi_i \tag{6.5}$$

These quantities π_i are typically the focus of analyses of Markov chains.

We will use the symbol π to name the column vector of all the π_i:

$$\pi = (\pi_1, \pi_2, ...)' \tag{6.6}$$

where $'$ means matrix transpose.

6.3.1 Stationary Distribution

The π_i are called **stationary probabilities**, because if the initial state X_0 is a random variable with that distribution, then all X_i will have that distribution. Here's why:

Using (6.5), we have

$$\begin{aligned} \pi_i &= \lim_{n\to\infty} P(X_n = i) && (6.7)\\ &= \lim_{n\to\infty} \sum_k P(X_{n-1} = k)\ p_{ki} && (6.8)\\ &= \sum_k \pi_k\ p_{ki} && (6.9) \end{aligned}$$

So, if $P(X_0 = i) = \pi_i$, then

[4] Basically, we need the chain to not be **periodic**. Consider a random walk, for instance: We start at position 0 on the number line, at time 0. The states are the integers. (So, this chain has an infinite state space.) At each time, we flip a coin to determine whether to move right (heads) or left (tails) 1 unit. A little thought shows that if we start at 0, the only times we can return to 0 are even-number times, i.e. $P(X_n = 0\ |X_0 = 0)$ for all odd numbers n. This is a periodic chain. By the way, (6.4) turns out to be 0 for this chain.

$$P(X_1 = i) = \sum_k P(X_0 = k)\ p_{ki} \tag{6.10}$$

$$= \sum_k \pi_k\ p_{ki} \tag{6.11}$$

$$= \pi_i \tag{6.12}$$

this last using (6.9). So, if X_0 has distribution π, then the same will be true for X_1, and contiuing in this manner we see that $X_2, X_3, ...$ will all have that distribution, thus demostrating the claimed stationary property for π. (This will be illustrated more concretely in Section 6.4.2.)

Of course, (6.7) holds for all states i. So in matrix terms, (6.7) says

$$\pi' = \pi' P \tag{6.13}$$

6.3.2 Calculation of π

Equation (6.13) then shows us how to find the π_i, at least in the case of finite state spaces, the subject of this section here, as follows.

First, rewrite (6.13)

$$(I - P')\pi = 0 \tag{6.14}$$

Here I is the $n \times n$ identity matrix (for a chain with n states), and again $'$ denotes matrix transpose.

This equation has infinitely many solutions; if π is a solution, then so for example is 8π. Moreover, the equation shows that the matrix there, $I - P'$, cannot have an inverse; if it did, we could multiply both sides by the inverse, and find that the unique solution is $\pi = 0$, which can't be right. This says in turn that the rows of $I - P'$ are not linearly independent.

The latter fact is quite important, for the following reason. Recall the close connection of matrix inverse and systems of linear equations. (6.14), a matrix equation, represents a system of n linear equations in n unknowns, the latter being $\pi_1, \pi_2, .., \pi_n$. So, the lack of linear independence of the rows of $I - P'$ means, in plain English, that at least one of those equations is redundant.

But we need n independent equations, and fortunately one is available:

$$\sum_i \pi_i = 1 \tag{6.15}$$

Note that (6.15) can be written as

$$O'\pi = 1 \tag{6.16}$$

where O is a vector of n 1s. Excellent, let's use it!

So, again, thinking of (6.14) as a system of linear equations, let's replace the last equation by (6.16). Switching back to the matrix view, that means that we replace the last row in the matrix $I - P'$ in (6.14) by O', and correspondingly replace the last element of the right-side vector by 1. Now we have a nonzero vector on the right side, and a full-rank (i.e. invertible) matrix on the left side. This is the basis for the following code, which we will use for finding π.

```
1  findpi1 <- function(p) {
2      n <- nrow(p)
3      # find I-P'
4      imp <- diag(n) - t(p)
5      # replace the last row of I-P' as discussed
6      imp[n,] <- rep(1,n)
7      # replace the corresponding element of the
8      # right side by (the scalar) 1
9      rhs <- c(rep(0,n-1),1)
10     # now use R's built-in solve()
11     solve(imp,rhs)
12 }
```

6.3.2.1 Example: π in Die Game

Consider the die game example above. Guess what! Applying the above code, all the π_i turn out to be 1/10. In retrospect, this should be obvious. If we were to draw the states 1 through 10 as a ring, with 1 following 10, it should be clear that all the states are completely symmetric.

6.3.2.2 Another Way to Find π

Here is another way to compute π. It is not commonly used, but **it will also help illustrate some of the concepts.**

Suppose (6.5) holds. Recall that P^m is the m-step transition matrix, so that for instance row 1 of that matrix is the set of probabilities of going from state 1 to the various states in m steps. The

same will be true for row 2 and so on. Putting that together with (6.5), we have that

$$\lim_{n\to\infty} P^n = \Pi \tag{6.17}$$

where the $n \times n$ matrix Π has each of its rows equal to π'.

We can use this to find π. We take P to a large power m, and then each of the rows will approximate π. In fact, we can get an even better appoximation by averaging the rows.

Moreover, we can save a lot of computation by noting the following. Say we want the 16^{th} power of P. We could set up a loop with 15 iterations, building up a product. But actually we can do it with just 4 iterations. We first square P, yielding P^2. But then we square *that*, yielding P^4. Square twice more, yielding P^8 and finally P^{16}. This is especially fast on a GPU (graphics processing unit).

```
# finds stationary probabilities of a Markov chain using matrix powers

altfindpi <- function(p,k) {
   niters <- ceiling(log2(k))
   prd <- p
   for (i in 1:niters) {
      prd <- prd %*% prd
   }
   colMeans(prd)
}
```

This approach has the advantage of being easy to parallelize, unlike matrix inversion.

6.4 Example: 3-Heads-in-a-Row Game

How about the following game? We keep tossing a coin until we get three consecutive heads. What is the expected value of the number of tosses we need?

We can model this as a Markov chain with states 0, 1, 2 and 3, where state i means that we have accumulated i consecutive heads so far. If we simply stop playing the game when we reach state 3, that state would be known as an **absorbing state**, one that we never leave.

We could proceed on this basis, but to keep things elementary, let's just model the game as being played repeatedly, as in the die game above. You'll see that that will still allow us to answer the original question. Note that now that we are taking that approach, it will suffice to have just three

states, 0, 1 and 2; there is no state 3, because as soon as we win, we immediately start a new game, in state 0.

6.4.1 Markov Analysis

Clearly we have transition probabilities such as p_{01}, p_{12}, p_{10} and so on all equal to 1/2. Note from state 2 we can only go to state 0, so $p_{20} = 1$.

Here's the code below. Of course, since R subscripts start at 1 instead of 0, we must recode our states as 1, 2 an 3.

```
p <- matrix(rep(0,9),nrow=3)
p[1,1] <- 0.5
p[1,2] <- 0.5
p[2,3] <- 0.5
p[2,1] <- 0.5
p[3,1] <- 1
findpi1(p)
```

It turns out that

$$\pi = (0.5714286, 0.2857143, 0.1428571) \tag{6.18}$$

So, in the long run, about 57.1% of our tosses will be done while in state 0, 28.6% while in state 1, and 14.3% in state 2.

Now, look at that latter figure. Of the tosses we do while in state 2, half will be heads, so half will be wins. In other words, about 0.071 of our tosses will be wins. And THAT figure answers our original question, through the following reasoning:

Think of, say, 10000 tosses. There will be about 710 wins sprinkled among those 10000 tosses. Thus the average number of tosses between wins will be about 10000/710 = 14.1. In other words, the expected time until we get three consecutive heads is about 14.1 tosses.

6.4.2 Back to the word "Stationary"

Let's look at the term *stationary distribution* in the context of this game.

At time 0, we haven't done any coin flips, so $X_0 = 0$. But consider a modified version of the game, in which at time 0 you are given "credit" or a "head start. For instance, you might be allowed to start in state 1, meaning that you already have one consecutive head, even though you actually haven't done any flips.

Now, suppose your head start is random, so that you start in state 0, 1 or 2 with certain probabilities. And suppose those probabilities are given by π! In other words, you start in state 0, 1 or 2, with probabilities 0.57, 0.29 and 0.14, as in (6.18). What, then, will be the distribution of X_1?

$$\begin{aligned} p_{X_1}(i) &= P(X_1 = i) && (6.19) \\ &= \sum_{j=0}^{2} P(X_0 = j)\ P(X_1 = i\ |X_0 = j) && (6.20) \\ &= \sum_{j=0}^{2} \pi_j\ p(j,i) && (6.21) \end{aligned}$$

This is exactly what we saw in (6.12) and the equations leading up to it. So we know that (6.21) will work out to π_i. In other words, $P(X_1 = 0) = 0.57$ and so on. (Work it out numerically if you are in doubt.) So, we see that

> If X_0 has distribution π, then the same will be true for X_1. We say that the distribution of the chain is *stationary*.

6.5 A Modified Notebook Analysis

Our previous notebook analysis (and most of our future ones, other than for Markov chains), relied on imagining performing many independent replications of the same experiment.

6.5.1 A Markov-Chain Notebook

Consider Table 2.3, for instance. There our experiment was to watch the network during epochs 1 and 2. So, on the first line of the notebook, we would watch the network during epochs 1 and 2 and record the result. On the second line, we watch a new, independent replication of the network during epochs 1 and 2, and record the results.

But instead of imagining a notebook recording infinitely many replications of the two epochs, we could imagine watching just *one* replication but over infinitely many epochs. We'd watch the network during epoch 1, epoch 2, epoch 3 and so on. Now one line of the notebook would record one epoch.

For general Markov chains, each line would record one time step. We would have columns of the notebook labeled n and X_n. The reason this approach would be natural is (6.4). In that context, π_i would be the long-run proportion of notebook lines in which $X_n = i$.

6.5.2 Example: 3-Heads-in-a-Row Game

For instance, consider the 3-heads-in-a-row game. Then (6.18) says that about 57% of the notebook lines would have a 0 in the X_n column, with about 29% and 14% of the lines showing 1 and 2, respectively.

Moreover, suppose we also have a notebook column labeled *Win*, with an entry Yes in a certain line meaning, yes, that coin flip resulted in a win, with a No entry meaning no win. Then the mean time until a win, which we found to be 14.1 above, would be described in notebook terms as the long-run average number of lines between Yes entries in the *Win* column.

6.6 Simulation of Markov Chains

Following up on Section 6.5, recall that our previous simulations have basically put into code form our notebook concept. Our simulations up until now have been based on the definition of probability, which we had way back in Section 2.3. Our simulation code modeled the notebook independent replications notion. We can do a similar thing now, based on the ideas in Section 6.5.

In a time series kind of situation such as Markov chains, since we are interested in long-run behavior in the sense of time, our simulation is based on (6.5). In other words, we simulate the evolution of X_n as n increases, and take long-run averages of whatever is of interest.

Here is simulation code for the example in Section 6.4, caculating the approximate value of the long-run time between wins (found to be about 14.1 by mathematical means above):

```
# simulation of 3-in-a-row coin toss gaem

threeinrow <- function(ntimesteps) {
   consec <- 0  # number of consec Hs
   nwins <- 0  # number of wins
   wintimes <- 0  # total of times to win
   startplay <- 0  # time step 0
   for (i in 1:ntimesteps) {
      if (toss() == 'H') {
         consec <- consec + 1
         if (consec == 3) {
            nwins <- nwins + 1
            wintimes <-
               wintimes + i - startplay
            consec <- 0
            startplay <- i
```

```
            }
        } else consec <- 0
    }
    wintimes / nwins
}

toss <- function()
    if (runif(1) < 0.5) 'H' else 'T'
```

l

6.7 Example: ALOHA

Consider our old friend, the ALOHA network model. (You may wish to review the statement of the model in Section 2.5 before continuing.) The key point in that system is that it was "memoryless," in that the probability of what happens at time k+1 depends only on the state of the system at time k.

For instance, consider what might happen at time 6 if $X_5 = 2$. Recall that the latter means that at the end of epoch 5, both of our two network nodes were active. The possibilities for X_6 are then

- X_6 will be 2 again, with probability $p^2 + (1-p)^2$
- X_6 will be 1, with probability $2p(1-p)$

The central point here is that the past history of the system—i.e. the values of $X_1, X_2, X_3,$ and X_4—don't have any impact. We can state that precisely:

> The quantity
>
> $$P(X_6 = j | X_1 = i_1, X_2 = i_2, X_3 = i_3, X_4 = i_4, X_5 = i) \tag{6.22}$$
>
> does not depend on $i_m, m = 1, ..., 4$. Thus we can write (6.22) simply as $P(X_6 = j | X_5 = i)$.

Furthermore, that probability is the same as $P(X_9 = j | X_8 = i)$ and in general $P(X_{k+1} = j | X_k = i)$. So, we do have a Markov chain.

Since this is a three-state chain, the p_{ij} form a 3x3 matrix:

$$P = \begin{pmatrix} (1-q)^2 + 2q(1-q)p & 2q(1-q)(1-p) + 2q^2p(1-p) & q^2[p^2 + (1-p)^2] \\ (1-q)p & 2qp(1-p) + (1-q)(1-p) & q[p^2 + (1-p)^2] \\ 0 & 2p(1-p) & p^2 + (1-p)^2 \end{pmatrix} \tag{6.23}$$

For instance, the element in row 0, column 2, p_{02}, is $q^2[p^2 + (1-p)^2]$, reflecting the fact that to go from state 0 to state 2 would require that both inactive nodes become active (which has probability q^2, and then either both try to send or both refrain from sending (probability $p^2 + (1-p)^2$.

For the ALOHA example here, with p = 0.4 and q = 0.3, the solution is $\pi_0 = 0.47$, $\pi_1 = 0.43$ and $\pi_2 = 0.10$.

So we know that in the long run, about 47% of the epochs will have no active nodes, 43% will have one, and 10% will have two. From this we see that the long-run average number of active nodes is

$$0 \cdot 0.47 + 1 \cdot 0.43 + 2 \cdot 0.10 = 0.63 \tag{6.24}$$

By the way, note that every row in a transition matrix must sum to 1. (The probability that we go from state i to *somewhere* is 1, after all, so we must have $\sum_j p_{ij} = 1$.) That implies that we can save some work in writing R code; the last column must be 1 minus the others. In our example above, we would write

```
transmat <- matrix(rep(0,9),nrow=3)
p1 <- 1 - p
q1 <- 1 - q
transmat[1,1] <- q1^2 + 2 * q * q1 *p
transmat[1,2] <- 2 * q * q1 * p1 + 2 * q^2 * p * p1
transmat[2,1] <- q1 * p
transmat[2,2] <- 2 * q * p * p1 + q1 * p1
transmat[3,1] <- 0
transmat[3,2] <- 2 * p * p1
transmat[,3] <- 1 - p[,1] - p[,2]
findpi1(transmat)
```

Note the vectorized addition and recycling (Section 2.14.2).

6.8 Example: Bus Ridership Problem

Consider the bus ridership problem in Section 2.12. Make the same assumptions now, but add a new one: There is a maximum capacity of 20 passengers on the bus.

The random variables L_i, i = 1,2,3,... form a Markov chain. Let's look at some of the transition probabilities:

$$p_{00} = 0.5 \tag{6.25}$$

$$p_{01} = 0.4 \tag{6.26}$$

$$p_{11} = (1 - 0.2) \cdot 0.5 + 0.2 \cdot 0.4 \tag{6.27}$$

$$p_{20} = (0.2)^2(0.5) = 0.02 \tag{6.28}$$

$$p_{20,20} = (0.8)^{20}(0.5 + 0.4 + 0.1) + \binom{20}{1}(0.2)^1(0.8)^{20-1}(0.4 + 0.1) + \binom{20}{2}(0.2)^2(0.8)^{18}(0.1) \tag{6.29}$$

(Note that for clarity, there is a comma in $p_{20,20}$, as p_{2020} would be confusing and in some other examples even ambiguous. A comma is not necessary in p_{11}, since there must be two subscripts; the 11 here can't be eleven.)

After finding the π vector as above, we can find quantities such as the long-run average number of passengers on the bus,

$$\sum_{i=0}^{20} \pi_i i \tag{6.30}$$

We can also compute the long-run average number of would-be passengers who fail to board the bus. Denote by A_i denote the number of passengers on the bus as it *arrives* at stop i. The key point is that since $A_i = L_{i-1}$, then (6.4) and (6.5) will give the same result, no matter whether we look at the L_j chain or the A_j chain.

Now, armed with that knowledge, let D_j denote the number of disappointed people at stop j. Then

$$ED_j = 1 \cdot P(D_j = 1) + 2 \cdot P(D_j = 2). \tag{6.31}$$

That latter probability, for instance, is

$$\begin{aligned} P(D_j = 2) &= P(A_j = 20 \text{ and } B_j = 2 \text{ and } G_j = 0) && (6.32) \\ &= P(A_j = 20)\ P(B_j = 2)\ P(G_j = 0) && (6.33) \\ &= P(A_j = 20) \cdot 0.1 \cdot 0.8^{20} && (6.34) \end{aligned}$$

where as before B_j is the number who wish to board at stop j, and G_j is the number who get off the bus at that stop. Using the same reasoning, one can find $P(D_j = 1)$. (A number of cases to consider, left as an exercise for the reader.)

Taking the limits as $j \to \infty$, we have the long-run average number of disappointed customers on the left, and on the right, the terms $P(A_j = 19)$ and $P(A_j = 20)$ go to π_{19} and π_{20}, which we have and thus can obtain the value regarding disappointed customers.

Let's find the long-run average number of customers who alight from the bus. This can be done by considering all the various cases, but (4.68) really shortens our work:

$$EG_n = \sum_{i=0}^{20} P(A_n = i) E(G_n | A_n = i) \tag{6.35}$$

Given $A_n = i$, G_n has a binomial distribution with i trials and success probability 0.2, so

$$E(G_n | A_n = i) = i \cdot 0.2 \tag{6.36}$$

So, the right-hand side of 6.35 converges to

$$\sum_{i=0}^{20} \pi_i i \cdot 0.2 \tag{6.37}$$

In other words, the long-run average number alighting is 0.2 times (6.30).

6.9 Example: an Inventory Model

Consider the following simple inventory model. A store has 1 or 2 customers for a certain item each day, with probabilities v and w (v+w = 1). Each customer is allowed to buy only 1 item.

When the stock on hand reaches 0 on a day, it is replenished to r items immediately after the store closes that day.

If at the start of a day the stock is only 1 item and 2 customers wish to buy the item, only one customer will complete the purchase, and the other customer will leave emptyhanded.

Let X_n be the stock on hand at the end of day n (*after* replenishment, if any). Then $X_1, X_2, ...$ form a Markov chain, with state space 1,2,...,r.

The transition probabilities are easy to find. Take p_{21}, for instance. If there is a stock of 2 items at the end of one day, what is the (conditional) probability that there is only 1 item at the end of the next day? Well, for this to happen, there would have to be just 1 customer coming in, not 2, and that has probability v. So, $p_{21} = v$. The same reasoning shows that $p_{2r} = w$.

Let's write a function **inventory(v,w,r)** that returns the π vector for this Markov chain. It will call **findpi1()**, similarly to the two code snippets on page 120. For convenience, let's assume **r** is at least 3.[5]

```
1  inventory <- function(v,w,r) {
2      tm <- matrix(rep(0,r^2),nrow=r)
3      for (i in 3:r) {
4          tm[i,i-1] <- v
5          tm[i,i-2] <- w
6      }
7      tm[2,1] <- v
8      tm[2,r] <- w
9      tm[1,r] <- 1
10     findpi1(tm)
11 }
```

6.10 Expected Hitting Times

Consider an n-state Markov chain that is *irreducible*, meaning that it is possible to get from any state i to any other state j in some number of steps. Define the random variable T_{ij} to be the time needed to go from state i to state j. (Note that T_{ii} is NOT 0, though it can be 1 if $p_{ii} > 0$.)

In many applications, we are interested in the mean time to reach a certain state, given that we start at some specified state. In other words, we wish to calculate ET_{ij}.

The material in Section 4.15 is useful here. In (4.68), take $V = T_{ij}$ and take U to be the first state

[5] If **r** is 2, then the expression 3:2 in the code evaluates to the vector (3,2), which would not be what we want in this case.

we visit after leaving state i (which could be i again). Then

$$E(T_{ij}) = \sum_k p_{ik} E(T_{ij} \mid U = k), \quad 1 \leq i, j \leq n \tag{6.38}$$

Consider what happens if $U = m \neq j$. Then we will have already used up 1 time step, and *still* will need an average of E_{Tkj} more steps to get to j. In other words,

$$E(T_{ij} \mid U = k \neq j) = 1 + ET_{kj} \tag{6.39}$$

Did you notice the Markov property being invoked?

On the other hand, the case $U = j$ is simple:

$$E(T_{ij} \mid U = j) = 1 \tag{6.40}$$

This then implies that

$$E(T_{ij}) = 1 + \sum_{k \neq j} p_{ik} E(T_{kj}), \quad 1 \leq i, j \leq n \tag{6.41}$$

We'll focus on the case $j = n$, i.e. look at how long it takes to get to state n. (We could of course do the same analysis for any other destination state.) Let η_i denote $E(T_{in})$, and define $\eta = (\eta_1, \eta_2, ..., \eta_{n-1})'$. (Note that η has only $n - 1$ components!) So,

$$\eta_i = 1 + \sum_{k<n} p_{ik} \eta_k, \quad 1 \leq i, j \leq n \tag{6.42}$$

Equation (6.42) is a system of linear equations, which we can write in matrix form. Then the code to solve the system is (remember, we have only an $(n-1) \times (n-1)$ system)

```
findeta <- function(p) {
    n <- nrow(p)
    q <- diag(n) - p
    q <- q[1:(n-1),1:(n-1)]
    ones <- rep(1,n-1)
    solve(q,ones)
}
```

Chapter 9

The Exponential Distributions

The family of exponential distributions, Section 7.6.3, has a number of remarkable properties, which contribute to its widespread usage in probabilistic modeling. We'll discuss those here.

9.1 Connection to the Poisson Distribution Family

Suppose the lifetimes of a set of light bulbs are independent and identically distributed **(i.i.d.)**, and consider the following process. At time 0, we install a light bulb, which burns an amount of time X_1. Then we install a second light bulb, with lifetime X_2. Then a third, with lifetime X_3, and so on.

Let

$$T_r = X_1 + ... + X_r \tag{9.1}$$

denote the time of the r^{th} replacement. Also, let N(t) denote the number of replacements up to and including time t. Then it can be shown that if the common distribution of the X_i is exponentially distributed, the N(t) has a Poisson distribution with mean λt. And the converse is true too: If the X_i are independent and identically distributed and N(t) is Poisson, then the X_i must have exponential distributions. In summary:

Theorem 17 *Suppose $X_1, X_2, ...$ are i.i.d. nonnegative continuous random variables. Define*

$$T_r = X_1 + ... + X_r \tag{9.2}$$

and

$$N(t) = \max\{k : T_k \leq t\} \tag{9.3}$$

Then the distribution of N(t) is Poisson with parameter λt for all t if and only if the X_i have an exponential distribution with parameter λ.

In other words, N(t) will have a Poisson distribution if and only if the lifetimes are exponentially distributed.

Proof

"Only if" part:

The key is to notice that the event $X_1 > t$ is exactly equivalent to $N(t) = 0$. If the first light bulb lasts longer than t, then the count of burnouts at time t is 0, and vice versa. Then

$$\begin{aligned} P(X_1 > t) &= P[N(t) = 0] \quad \text{(see above equiv.)} && (9.4)\\ &= \frac{(\lambda t)^0}{0!} \cdot e^{-\lambda t} \quad ((4.41) && (9.5)\\ &= e^{-\lambda t} && (9.6) \end{aligned}$$

Then

$$f_{X_1}(t) = \frac{d}{dt}(1 - e^{-\lambda t}) = \lambda e^{-\lambda t} \tag{9.7}$$

That shows that X_1 has an exponential distribution, and since the X_i are i.i.d., that implies that all of them have that distribution.

"If" part:

We need to show that if the X_i are exponentially distributed with parameter λ, then for u nonnegative and each positive integer k,

$$P[N(u) = k] = \frac{(\lambda u)^k e^{-\lambda u}}{k!} \tag{9.8}$$

The proof for the case k = 0 just reverses (9.4) above. The general case, not shown here, notes that $N(u) \leq k$ is equivalent to $T_{k+1} > u$. The probability of the latter event can be found by integrating

(7.46) from u to infinity. One needs to perform k-1 integrations by parts, and eventually one arrives at (9.8), summed from 1 to k, as required.

■

The collection of random variables N(t) $t \geq 0$, is called a **Poisson process**.

The relation $E[N(t)] = \lambda t$ says that replacements are occurring at an average rate of λ per unit time. Thus λ is called the **intensity parameter** of the process. It is this "rate" interpretation that makes λ a natural indexing parameter in (7.40).

9.2 Memoryless Property of Exponential Distributions

One of the reasons the exponential family of distributions is so famous is that it has a property that makes many practical stochastic models mathematically tractable: The exponential distributions are **memoryless**.

9.2.1 Derivation and Intuition

What the term *memoryless* means for a random variable W is that for all positive t and u

$$P(W > t + u | W > t) = P(W > u) \tag{9.9}$$

Any exponentially distributed random variable has this property. Let's derive this:

$$\begin{aligned} P(W > t + u | W > t) &= \frac{P(W > t + u \text{ and } W > t)}{P(W > t)} && (9.10)\\ &= \frac{P(W > t + u)}{P(W > t)} && (9.11)\\ &= \frac{\int_{t+u}^{\infty} \lambda e^{-\lambda s}\, ds}{\int_{t}^{\infty} \lambda e^{-\lambda s}\, ds} && (9.12)\\ &= e^{-\lambda u} && (9.13)\\ &= P(W > u) && (9.14) \end{aligned}$$

We say that this means that "time starts over" at time t, or that W "doesn't remember" what happened before time t.

It is difficult for the beginning modeler to fully appreciate the memoryless property. Let's make it concrete. Consider the problem of waiting to cross the railroad tracks on Eighth Street in Davis, just west of J Street. One cannot see down the tracks, so we don't know whether the end of the train will come soon or not.

If we are driving, the issue at hand is whether to turn off the car's engine. If we leave it on, and the end of the train does not come for a long time, we will be wasting gasoline; if we turn it off, and the end does come soon, we will have to start the engine again, which also wastes gasoline. (Or, we may be deciding whether to stay there, or go way over to the Covell Rd. railroad overpass.)

Suppose our policy is to turn off the engine if the end of the train won't come for at least s seconds. Suppose also that we arrived at the railroad crossing just when the train first arrived, and we have already waited for r seconds. Will the end of the train come within s more seconds, so that we will keep the engine on? If the length of the train were exponentially distributed (if there are typically many cars, we can model it as continuous even though it is discrete), Equation (9.9) would say that the fact that we have waited r seconds so far is of no value at all in predicting whether the train will end within the next s seconds. The chance of it lasting at least s more seconds right now is no more and no less than the chance it had of lasting at least s seconds when it first arrived.

9.2.2 Uniquely Memoryless

By the way, the exponential distributions are the only continuous distributions which are memoryless. (Note the word *continuous*; in the discrete realm, the family of geometric distributions are also uniquely memoryless.) This too has implications for the theory. A rough proof of this uniqueness is as follows:

Suppose some continuous random variable V has the memoryless property, and let R(t) denote $1 - F_V(t)$. Then from (9.9), we would have

$$R(t+u)]/R(t) = R(u) \tag{9.15}$$

or

$$R(t+u) = R(t)R(u) \tag{9.16}$$

Differentiating both sides with respect to t, we'd have

$$R\prime(t+u) = R\prime(t)R(u) \tag{9.17}$$

Setting t to 0, this would say

$$R\prime(u) = R\prime(0)R(u) \tag{9.18}$$

This is a well-known differential equation, whose solution is

$$R(u) = e^{-cu} \tag{9.19}$$

which is exactly 1 minus the cdf for an exponentially distributed random variable.

9.2.3 Example: "Nonmemoryless" Light Bulbs

Suppose the lifetimes in years of light bulbs have the density 2t/15 on (1,4), 0 elsewhere. Say I've been using bulb A for 2.5 years now in a certain lamp, and am continuing to use it. But at this time I put a new bulb, B, in a second lamp. I am curious as to which bulb is more likely to burn out within the next 1.2 years. Let's find the two probabilities.

For bulb A:

$$P(L > 3.7|L > 2.5) = \frac{P(L > 3.7)}{P(L > 2.5)} = 0.24 \tag{9.20}$$

For bulb B:

$$P(X > 1.2) = \int_{1.2}^{4} 2t/15 \; dt = 0.97 \tag{9.21}$$

So you can see that the bulbs do have "memory." We knew this beforehand, since the exponential distributions are the only continuous ones that have no memory.

9.3 Example: Minima of Independent Exponentially Distributed Random Variables

The memoryless property of the exponential distribution (Section 9.2 leads to other key properties. Here's a famous one:

Theorem 18 *Suppose $W_1, ..., W_k$ are independent random variables, with W_i being exponentially distributed with parameter λ_i. Let $Z = \min(W_1, ..., W_k)$. Then Z too is exponentially distributed with parameter $\lambda_1 + ... + \lambda_k$, and thus has mean equal to the reciprocal of that parameter*

Comments:

- In "notebook" terms, we would have k+1 columns, one each for the W_i and one for Z. For any given line, the value in the Z column will be the smallest of the values in the columns for $W_1, ..., W_k$; Z will be equal to one of them, but not the same one in every line. Then for instance $P(Z = W_3)$ is interpretable in notebook form as the long-run proportion of lines in which the Z column equals the W_3 column.
- It's pretty remarkable that the minimum of independent exponential random variables turns out again to be exponential. Contrast that with Section 17.3.6, where it is found that the minimum of independent uniform random variables does NOT turn out to have a uniform distribution.
- The sum $\lambda_1 + ... + \lambda_n$ in (a) should make good intuitive sense to you, for the following reasons. Recall from Section 9.1 that the parameter λ in an exponential distribution is interpretable as a "light bulb burnout rate."

  Say we have persons 1 and 2. Each has a lamp. Person i uses Brand i light bulbs, i = 1,2. Say Brand i light bulbs have exponential lifetimes with parameter λ_i. Suppose each time person i replaces a bulb, he shouts out, "New bulb!" and each time *anyone* replaces a bulb, I shout out "New bulb!" Persons 1 and 2 are shouting at a rate of λ_1 and λ_2, respectively, so I am shouting at a rate of $\lambda_1 + \lambda_2$.

Proof

$$
\begin{aligned}
F_Z(t) &= P(Z \leq t) \quad \text{(def. of cdf)} && (9.22)\\
&= 1 - P(Z > t) && (9.23)\\
&= 1 - P(W_1 > t \text{ and ... and } W_k > t) \quad (\min > t \text{ iff all } W_i > t) && (9.24)\\
&= 1 - \Pi_i \; P(W_i > t) \quad \text{(indep.)} && (9.25)\\
&= 1 - \Pi_i \; e^{-\lambda_i t} \quad \text{(expon. distr.)} && (9.26)\\
&= 1 - e^{-(\lambda_1 + ... + \lambda_n)t} && (9.27)
\end{aligned}
$$

Taking $\frac{d}{dt}$ of both sides proves the theorem.

■

Also:

Theorem 19 *Under the conditions in Theorem 18,*

$$P(W_i < W_1, ..., W_{i-1}, W_{i+1}, ..., W_k) = \frac{\lambda_i}{\lambda_1 + ... + \lambda_k} \tag{9.28}$$

(There are k terms in the denominator, not k-1.)

Equation (9.28) should be intuitively clear as well from the above "thought experiment" (in which we shouted out "New bulb!"): On average, we have one new Brand 1 bulb every $1/\lambda_1$ time, so in a long time t, we'll have about $t\lambda_1$ shouts for this brand. We'll also have about $t\lambda_2$ shouts for Brand 2. So, a proportion of about

$$\frac{t\lambda_1}{t\lambda_1 + t\lambda_2} \tag{9.29}$$

of the shots are for Brand 1. Also, at any given time, the memoryless property of exponential distributions implies that the time at which I shout next will be the *minimum* of the times at which persons 1 and 2 shout next. This intuitively implies (9.28).

Proof

Again consider the case k = 2, and then use induction.

Let $Z = \min(W_1, W_2)$ as before. Then

$$P(Z = W_1 | W_1 = t) = P(W_2 > t \mid W_1 = t) \tag{9.30}$$

(Note: We are working with continuous random variables here, so quantities like $P(W_1 = t)$ are 0 (though actually $P(Z = W_1)$ is nonzero). So, as mentioned in Section 7.78, quantities like $P(Z = W_1 \mid W_1 = t)$ really mean "the probability that $W_2 > t$ in the conditional distribution of Z given W_1.)

Since W_1 and W_2 are independent,

$$P(W_2 > t \mid W_1 = t) = P(W_2 > t) = e^{-\lambda_2 t} \tag{9.31}$$

Now use (7.78):

$$P(Z = W_1) = \int_0^\infty \lambda_1 e^{-\lambda_1 t} e^{-\lambda_2 t} \, dt = \frac{\lambda_1}{\lambda_1 + \lambda_2} \tag{9.32}$$

as claimed.

■

This property of minima of independent exponentially-distributed random variables developed in this section is key to the structure of continuous-time Markov chains, in Chapter 11.

9.3.1 Example: Computer Worm

A computer science graduate student at UCD, Senthilkumar Cheetancheri, was working on a worm alert mechanism. A simplified version of the model is that network hosts are divided into groups of size g, say on the basis of sharing the same router. Each infected host tries to infect all the others in the group. When g-1 group members are infected, an alert is sent to the outside world.

The student was studying this model via simulation, and found some surprising behavior. No matter how large he made g, the mean time until an external alert was raised seemed bounded. He asked me for advice.

I modeled the nodes as operating independently, and assumed that if node A is trying to infect node B, it takes an exponentially-distributed amount of time to do so. This is a continuous-time Markov chain. Again, this topic is much more fully developed in Chapter 11, but all we need here is the result of Section 9.3, that exponential distributions are "memoryless.".

In state i, there are i infected hosts, each trying to infect all of the g-i noninfected hosts. When the process reaches state g-1, the process ends; we call this state an **absorbing state**, i.e. one from which the process never leaves.

Scale time so that for hosts A and B above, the mean time to infection is 1.0. Since in state i there are i(g-i) such pairs, the time to the next state transition is the minimum of i(g-i) exponentially-distributed random variables with mean 1. Theorem 18 tells us that this minimum is also exponentially distributed, with parameter $i(g-i) \cdot 1$. Thus the mean time to go from state i to state i+1 is 1/[i(g-i)].

Then the mean time to go from state 1 to state g-1 is

$$\sum_{i=1}^{g-1} \frac{1}{i(g-i)} \tag{9.33}$$

Using a calculus approximation, we have

$$\int_1^{g-1} \frac{1}{x(g-x)}\, dx = \frac{1}{g} \int_1^{g-1} (\frac{1}{x} + \frac{1}{g-x})\, dx = \frac{2}{g} \ln(g-1) \tag{9.34}$$

The latter quantity goes to zero as $g \to \infty$. This confirms that the behavior seen by the student in simulations holds in general. In other words, (9.33) remains bounded as $g \to \infty$. This is a very interesting result, since it says that the mean time to alert is bounded no matter how big our group size is.

So, even though our model here was quite simple, probably overly so, it did explain why the student was seeing the surprising behavior in his simulations.

9.3.2 Example: Electronic Components

Suppose we have three electronic parts, with independent lifetimes that are exponentially distributed with mean 2.5. They are installed simultaneously. Let's find the mean time until the last failure occurs.

Actually, we can use the same reasoning as for the computer worm example in Section 9.3.1: The mean time is simply

$$1/(3 \cdot 0.4) + 1/(2 \cdot 0.4) + 1/(1 \cdot 0.4) \tag{9.35}$$

9.4 A Cautionary Tale: the Bus Paradox

Suppose you arrive at a bus stop, at which buses arrive according to a Poisson process with intensity parameter 0.1, i.e. 0.1 arrival per minute. Recall that the means that the interarrival times have an exponential distribution with mean 10 minutes. What is the expected value of your waiting time until the next bus?

Well, our first thought might be that since the exponential distribution is memoryless, "time starts over" when we reach the bus stop. Therefore our mean wait should be 10.

On the other hand, we might think that on average we will arrive halfway between two consecutive buses. Since the mean time between buses is 10 minutes, the halfway point is at 5 minutes. Thus it would seem that our mean wait should be 5 minutes.

Which analysis is correct? Actually, the correct answer is 10 minutes. So, what is wrong with the second analysis, which concluded that the mean wait is 5 minutes? The problem is that the second analysis did not take into account the fact that although inter-bus intervals have an exponential distribution with mean 10, *the particular inter-bus interval that we encounter is special.*

9.4.1 Length-Biased Sampling

Imagine a bag full of sticks, of different lengths. We reach into the bag and choose a stick at random. The key point is that not all pieces are equally likely to be chosen; the longer pieces will have a greater chance of being selected.

Say for example there are 50 sticks in the bag, with ID numbers from 1 to 50. Let X denote the length of the stick we obtain if select a stick on an equal-probability basis, i.e. each stick having probability 1/50 of being chosen. (We select a random number I from 1 to 50, and choose the stick with ID number I.) On the other hand, let Y denote the length of the stick we choose by reaching into the bag and pulling out whichever stick we happen to touch first. Intuitively, the distribution of Y should favor the longer sticks, so that for instance $EY > EX$.

Let's look at this from a "notebook" point of view. We pull a stick out of the bag by random ID number, and record its length in the X column of the first line of the notebook. Then we replace the stick, and choose a stick out by the "first touch" method, and record its length in the Y column of the first line. Then we do all this again, recording on the second line, and so on. Again, because the "first touch" method will favor the longer sticks, the long-run average of the Y column will be larger than the one for the X column.

Another example was suggested to me by UCD grad student Shubhabrata Sengupta. Think of a large parking lot on which hundreds of buckets are placed of various diameters. We throw a ball high into the sky, and see what size bucket it lands in. Here the density would be proportional to area of the bucket, i.e. to the square of the diameter.

Similarly, the particular inter-bus interval that we hit is likely to be a longer interval. To see this, suppose we observe the comings and goings of buses for a very long time, and plot their arrivals on a time line on a wall. In some cases two successive marks on the time line are close together, sometimes far apart. If we were to stand far from the wall and throw a dart at it, we would hit the interval between some pair of consecutive marks. Intuitively we are more apt to hit a wider interval than a narrower one.

The formal name for this is **length-biased sampling**.

Once one recognizes this and carefully derives the density of that interval (see below), we discover that that interval does indeed tend to be longer—so much so that the expected value of this interval is 20 minutes! Thus the halfway point comes at 10 minutes, consistent with the analysis which appealed to the memoryless property, thus resolving the "paradox."

In other words, if we throw a dart at the wall, say, 1000 times, the mean of the 1000 intervals we would hit would be about 20. This in contrast to the mean of all of the intervals on the wall, which would be 10.

9.4.2 Probability Mass Functions and Densities in Length-Biased Sampling

Actually, we can intuitively reason out what the density is of the length of the particular inter-bus interval that we hit, as follows.

First consider the bag-of-sticks example, and suppose (somewhat artificially) that stick length X is a discrete random variable. Let Y denote the length of the stick that we pick by randomly touching a stick in the bag.

Again, note carefully that for the reasons we've been discussing here, the distributions of X and Y are different. Say we have a list of all sticks, and we choose a stick at random from the list. Then the length of that stick will be X. But if we choose by touching a stick in the bag, that length will be Y.

Now suppose that, say, stick lengths 2 and 6 each comprise 10% of the sticks in the bag, i.e.

$$p_X(2) = p_X(6) = 0.1 \tag{9.36}$$

Intuitively, one would then reason that

$$p_Y(6) = 3p_Y(2) \tag{9.37}$$

In other words, even though the sticks of length 2 are just as numerous as those of length 6, the latter are three times as long, so they should have triple the chance of being chosen. So, the chance of our choosing a stick of length j depends not only on $p_X(j)$ but also on j itself.

We could write that formally as

$$p_Y(j) \propto j p_X(j) \tag{9.38}$$

where $\propto$ is the "is proportional to" symbol. Thus

$$p_Y(j) = cjp_X(j) \tag{9.39}$$

for some constant of proportionality c.

But a probability mass function must sum to 1. So, summing over all possible values of j (whatever they are), we have

$$1 = \sum_j p_Y(j) = \sum_j cjp_X(j) \tag{9.40}$$

That last term is c E(X)! So, c = 1/EX, and

$$p_Y(j) = \frac{1}{EX} \cdot jp_X(j) \tag{9.41}$$

The continuous analog of (9.41) is

$$f_Y(t) = \frac{1}{EX} \cdot tf_X(t) \tag{9.42}$$

So, for our bus example, in which $f_X(t) = 0.1e^{-0.1t},\ t > 0$ and EX = 10,

$$f_Y(t) = 0.01te^{-0.1t} \tag{9.43}$$

You may recognize this as an Erlang density with r = 2 and $\lambda = 0.1$. That distribution does indeed have mean 20, consistent with the discussion at the end of Section 9.4.1.

Chapter 10

Stop and Review: Probability Structures

There's quite a lot of material in the preceding chapters, but it's crucial that you have a good command of it before proceeding, as the coming chapters will continue to build on it.

With that aim, here are the highlights of what we've covered so far, with links to the places at which they were covered:

- **expected value** (Section 3.5):

  Consider random variables X and Y (not assumed independent), and constants c_1 and c_2. We have:

  $$E(X+Y) = EX + EY \tag{10.1}$$

  $$E(c_1 X) = c_1 EX \tag{10.2}$$

  $$E(c_1 X + c_2 Y) = c_1 EX + c_2 EY \tag{10.3}$$

  By induction,

  $$E(a_1 U_1 + ... + a_k U_k) = a_1 EX_1 + ... + a_k EX_k \tag{10.4}$$

  for random variables U_i and constants a_i.

- **variance** (Section 3.6):

  For any variable W,

$$Var(W) = E[(W - EW)^2] = E(W^2) - (EW)^2 \tag{10.5}$$

  Consider random variables X and Y (now assumed independent), and constants c_1 and c_2. We have:

$$Var(X + Y) = Var(X) + Var(Y) \tag{10.6}$$

$$Var(c_1 X) = c_1^2 Var(X) \tag{10.7}$$

  By induction,

$$Var(a_1 U_1 + ... + a_k U_k) = a_1^2 Var(U_1) + ... + a_k^2 Var(U_k) \tag{10.8}$$

  for independent random variables U_i and constants a_i.

- **indicator random variables** (Section 3.9):

  Equal 1 or 0, depending on whether a specified event A occurs.

  If T is an indicator random variable for the event A, then

$$ET = P(A), \quad Var(T) = P(A)[1 - P(A)] \tag{10.9}$$

- **distributions**:

  - **cdfs**(Section 7.3):
    For any random variable X,

$$F_X(t) = P(X \leq t), \qquad -\infty < t < \infty \tag{10.10}$$

  - **pmfs** (Section 3.13):
    For a discrete random variable X,

$$p_X(k) = P(X = k) \tag{10.11}$$

– **density functions** (Section 3.13):
For a continuous random variable X,

$$f_X(t) = \frac{d}{dt} F_X(t), \qquad -\infty < t < \infty \tag{10.12}$$

and

$$P(X \text{ in } A) = \int_A f_X(s)\ ds \tag{10.13}$$

- **famous parametric families of distributions**:

Just like one can have a family of curves, say $sin(2\pi n\theta(t)$ (different curve for each n and θ), certain families of distributions have been found useful. They're called *parametric families*, because they are indexed by one or more parameters, anlagously to n and θ above.

discrete:

– **geometric** (Section 4.2)
Number of i.i.d. trials until first success. For success probability p:

$$p_N(k) = (1-p)^k p \tag{10.14}$$

$$EN = 1/p, \quad Var(N) = \frac{1-p}{p^2} \tag{10.15}$$

– **binomial** (Section 4.3):
Number of successes in n i.i.d. trials, probability p of success per trial:

$$p_N(k) = \binom{n}{k} p^k (1-p)^{n-k} \tag{10.16}$$

$$EN = np, \quad Var(N) = np(1-p) \tag{10.17}$$

– **Poisson** (Section 4.5):
Has often been found to be a good model for counts over time periods.
One parameter, often called λ. Then

$$p_N(k) = \frac{e^{-\lambda}\lambda^k}{k!}, k = 0, 1, 2, ... \tag{10.18}$$

$$EN = Var(N) = \lambda \tag{10.19}$$

– **negative binomial** (Section 4.4):
Number of i.i.d. trials until rth success. For success probability p:

$$p_N(k) = \binom{k-1}{r-1}(1-p)^{k-r}p^r, k = r, r+1, ... \tag{10.20}$$

$$E(N) = r \cdot \frac{1}{p}, \quad Var(N) = r \cdot \frac{1-p}{p^2} \tag{10.21}$$

continuous:

– **uniform** (Section 7.6.1.1):
All points "equally likely." If the interval is (q,r),

$$f_X(t) = \frac{1}{r-q}, \quad q < t < r \tag{10.22}$$

$$EX == \frac{q+r}{2}, \quad Var(D) = \frac{1}{12}(r-q)^2 \tag{10.23}$$

– **normal (Gaussian)** (Section 7.6.2):
"Bell-shaped curves." Useful due to Central Limit Theorem (Section 8.9. (Thus good approximation to binomial distribution.)
Closed under affine transformations (Section 8.1.1)!
Parameterized by mean and variance, μ and σ^2:

$$f_X(t) = \frac{1}{\sqrt{2\pi}\sigma}\ e^{-0.5\left(\frac{t-\mu}{\sigma}\right)^2}, -\infty < t < \infty \tag{10.24}$$

exponential (Section 7.6.3):

– Memoryless! One parameter, usually called λ. Connectedt to Poisson family.

$$f_X(t) == \lambda e^{-\lambda t}, 0 < t < \infty \tag{10.25}$$

$$EX = 1/\lambda, \quad Var(X) = 1/\lambda^2 \tag{10.26}$$

– **gamma** (Section 7.6.4):
Special case, Erlang family, arises as the distribution of the sum of i.i.d. exponential random variables.

$$f_X(t) = \frac{1}{\Gamma(r)}\lambda^r t^{r-1}e^{-\lambda t},\ t > 0 \tag{10.27}$$

- **iterated expected values:**
  - For discrete U (4.68),

$$E(V) = \sum_c P(U = c)\ E(V \mid U = c) \tag{10.28}$$

  - For continuous V (7.77),

$$E(W) = \int_{-\infty}^{\infty} f_V(t)\ E(W \mid V = t)\ dt \tag{10.29}$$

Chapter 11

Introduction to Continuous-Time Markov Chains

In the Markov chains we analyzed in Chapter 6, events occur only at integer times. However, many Markov chain models are of the **continuous-time** type, in which events can occur at any times. Here the **holding time**, i.e. the time the system spends in one state before changing to another state, is a continuous random variable.

11.1 Continuous-Time Markov Chains

The state of a Markov chain at any time now has a continuous subscript. Instead of the chain consisting of the random variables $X_n,\ n = 1, 2, 3, ...$, it now consists of $\{X_t : t \in [0, \infty)\}$. The Markov property is now

$$P(X_{t+u} = k | X_s \text{ for all } 0 \leq s \leq t) = P(X_{t+u} = k | X_t) \text{ for all } t, u \geq 0 \tag{11.1}$$

11.2 Holding-Time Distribution

In order for the Markov property to hold, the distribution of holding time at a given state needs to be "memoryless." Recall that exponentially distributed random variables have this property. In other words, if a random variable W has density

$$f(t) = \lambda e^{-\lambda t} \tag{11.2}$$

for some λ then

$$P(W > r + s|W > r) = P(W > s) \tag{11.3}$$

for all positive r and s. And exponential distributions are the only continuous distributions which have this property. Therefore, *holding times in Markov chains must be exponentially distributed.*

Because it is central to the Markov property, the exponential distribution is assumed for all basic activities in Markov models. In queuing models, for instance, both the interarrival time and service time are assumed to be exponentially distributed (though of course with different values of λ). In reliability modeling, the lifetime of a component is assumed to have an exponential distribution.

Such assumptions have in many cases been verified empirically. If you go to a bank, for example, and record data on when customers arrive at the door, you will find the exponential model to work well (though you may have to restrict yourself to a given time of day, to account for nonrandom effects such as heavy traffic at the noon hour). In a study of time to failure for airplane air conditioners, the distribution was also found to be well fitted by an exponential density. On the other hand, in many cases the distribution is not close to exponential, and purely Markovian models cannot be used for anything more than a rough approximation.

11.2.1 The Notion of "Rates"

A key point is that the parameter λ in (11.2) has the interpretation of a rate, in the sense discussed in Theorem 18. To review, first, recall that $1/\lambda$ is the mean. Say light bulb lifetimes have an exponential distribution with mean 100 hours, so $\lambda = 0.01$. In our lamp, whenever its bulb burns out, we immediately replace it with a new on. Imagine watching this lamp for, say, 100,000 hours. During that time, we will have done approximately $100000/100 = 1000$ replacements. That would be using 1000 light bulbs in 100000 hours, so we are using bulbs at the rate of 0.01 bulb per hour. For a general λ, we would use light bulbs at the rate of λ bulbs per hour. This concept is crucial to what follows.

11.3 Stationary Distribution

In analogy to (6.4), we again define π_i to be the long-run proportion of time the system is in state i, where now N_{it} is the proportion of the time spent in state i, during (0,t). We again will derive a system of linear equations to solve for these proportions, using a flow out = flow in argument.

11.3.1 Intuitive Derivation

To this end, let λ_i denote the parameter in the holding-time distribution at state i, and define the quantities

$$\rho_{rs} = \lambda_r p_{rs} \tag{11.4}$$

where p_{rs} is that probably that, when a jump out of state r occurs, the jump is to state s.

The equations has the following interpretation. Note:

- λ_r is the rate of jumps out of state r, so
- $\lambda_r p_{rs}$ is the rate of jumps from state r to state s, and since
- π_r is the long-run proportion of the time we are in state r, then
- $\pi_r \lambda_r p_{rs}$ is the rate of jumps from r to s

Then, equating the rate of transitions into i and the rate out of i, we have

$$\pi_i \lambda_i = \sum_{j \neq i} \pi_j \lambda_j p_{ji} \tag{11.5}$$

These equations can then be solved for the π_i.

11.3.2 Computation

Motivated by (11.5), define the matrix Q by

$$q_{ij} = \begin{cases} \lambda_j p_{ji}, & \text{if } i \neq j \\ -\lambda_i, & \text{if } i = j \end{cases} \tag{11.6}$$

Q is called the **infinitesimal generator** of the system, so named because it is the basis of the system of differential equations that can be used to find the finite-time probabilistic behavior of X_t.

Then (11.5) is stated in matrix form as

$$Q\pi = 0 \tag{11.7}$$

But the π_i must sum to 1, so the above equation is subject to

$$1'\pi = 1 \tag{11.8}$$

where 1 denotes a (column) vector of n 1s, where n is the number of states.

In view of (11.8), the system (11.7) is redundant; there are n equations for n-1 unknowns, So, replace the last equation in (11.7) by (11.8).

Here is R code to solve the system:

```
findpicontin <- function(q) {
    n <- nrow(q)
    q[n,] <- rep(1,n)
    rhs <- c(rep(0,n-1),1)
    pivec <- solve(q,rhs)
    return(pivec)
}
```

To formulate the equations (11.5), we'll need a property of exponential distributions derived in Section 9.3, copied here for convenience:

Theorem 20 *Suppose $W_1, ..., W_k$ are independent random variables, with W_i being exponentially distributed with parameter λ_i. Let $Z = \min(W_1, ..., W_k)$. Then*

(a) Z is exponentially distributed with parameter $\lambda_1 + ... + \lambda_k$

(b) $P(Z = W_i) = \frac{\lambda_i}{\lambda_1 + ... + \lambda_k}$

11.4 Example: Machine Repair

Suppose the operations in a factory require the use of a certain kind of machine. The manager has installed two of these machines. This is known as a **gracefully degrading system**: When both machines are working, the fact that there are two of them, instead of one, leads to a shorter wait time for access to a machine. When one machine has failed, the wait is longer, but at least the factory operations may continue. Of course, if both machines fail, the factory must shut down until at least one machine is repaired.

Suppose the time until failure of a single machine, carrying the full load of the factory, has an exponential distribution with mean 20.0, but the mean is 25.0 when the other machine is working, since it is not so loaded. Repair time is exponentially distributed with mean 8.0.

We can take as our state space {0,1,2}, where the state is the number of working machines. Now, let us find the parameters λ_i and p_{ji} for this system. For example, what about λ_2? The holding time in state 2 is the minimum of the two lifetimes of the machines, and thus from the results of Section 9.3, has parameter $\frac{1}{25.0} + \frac{1}{25.0} = 0.08$.

For λ_1, a transition out of state 1 will be either to state 2 (the down machine is repaired) or to state 0 (the up machine fails). The time until transition will be the minimum of the lifetime of the up machine and the repair time of the down machine, and thus will have parameter $\frac{1}{20.0} + \frac{1}{8.0} = 0.175$. Similarly, $\lambda_0 = \frac{1}{8.0} + \frac{1}{8.0} = 0.25$.

It is important to understand how the Markov property is being used here. Suppose we are in state 1, and the down machine is repaired, sending us into state 2. Remember, the machine which had already been up has "lived" for some time now. But the memoryless property of the exponential distribution implies that this machine is now "born again."

What about the parameters p_{ji}? Well, p_{21} is certainly easy to find; since the transition $2 \rightarrow 1$ is the *only* transition possible out of state 2, $p_{21} = 1$.

For p_{12}, recall that transitions out of state 1 are to states 0 and 2, with rates 1/20.0 and 1/8.0, respectively. So,

$$p_{12} = \frac{1/8.0}{1/20.0 + 1/8.0} = 0.72 \tag{11.9}$$

Working in this manner, we finally arrive at the complete system of equations (11.5):

$$\pi_2(0.08) = \pi_1(0.125) \tag{11.10}$$

$$\pi_1(0.175) = \pi_2(0.08) + \pi_0(0.25) \tag{11.11}$$

$$\pi_0(0.25) = \pi_1(0.05) \tag{11.12}$$

In matrix terms:

$$\begin{pmatrix} -0.25 & 0.05 & 0 \\ 0.25 & -0.175 & 0.08 \\ 0 & 0.125 & -0.08 \end{pmatrix} \pi = 0 \tag{11.13}$$

Let's find the solution:

```
> q
      [,1]   [,2]  [,3]
[1,] -0.25  0.050  0.00
```

```
[2,]   0.25  -0.175   0.08
[3,]   0.00   0.125  -0.08
> findpicontin(q)
[1] 0.07239819 0.36199095 0.56561086
```

So,

$$\pi = (0.072, 0.362, 0.566) \tag{11.14}$$

Thus for example, during 7.2% of the time, there will be no machine available at all.

Several variations of this problem could be analyzed. We could compare the two-machine system with a one-machine version. It turns out that the proportion of down time (i.e. time when no machine is available) increases to 28.6%. Or we could analyze the case in which only one repair person is employed by this factory, so that only one machine can be repaired at a time, compared to the situation above, in which we (tacitly) assumed that if both machines are down, they can be repaired in parallel. We leave these variations as exercises for the reader.

11.5 Example: Migration in a Social Network

The following is a simplified version of research in online social networks.

There is a town with two social groups, with each person being in exactly one group. People arrive from outside town, with exponentially distributed interarrival times at rate α, and join one of the groups with probability 0.5 each. Each person will occasionally switch groups, with one possible "switch" being to leave town entirely. A person's time before switching is exponentially distributed with rate σ; the switch will either be to the other group or to the outside world, with probabilities q and 1-q, respectively. Let the state of the system be (i,j), where i and j are the number of current members in groups 1 and 2, respectively.

Let's find a typical balance equation, say for the state (8,8):

$$\pi_{(8,8)}(\alpha + 16 \cdot \sigma) = (\pi_{(9,8)} + \pi_{(8,9)}) \cdot 9\sigma(1-q) + (\pi_{(9,7)} + \pi_{(7,9)}) \cdot 9\sigma q + (\pi_{(8,7)} + \pi_{(7,8)}) \cdot 0.5\alpha \tag{11.15}$$

The reasoning is straightforward. How can we move out of state (8,8)? Well, there could be an arrival (rate α), or any one of the 16 people could switch groups (rate 16σ), etc.

Now, in a "going beyond finding the π" vein, let's find the long-run fraction of transfers into group 1 that come from group 2, as opposed to from the outside.

The rate of transitions into that group from outside is 0.5α. When the system is in state (i,j), the rate of transitions into group 1 from group 2 is $j\sigma q$, so the overall rate is $\sum_{i,j} \pi_{(i,j)} j\sigma q$. Thus the fraction of new members coming in to group 1 from transfers is

$$\frac{\sum_{i,j} \pi_{(i,j)} j\sigma q}{0.5\alpha + \sum_{i,j} \pi_{(i,j)} j\sigma q} \tag{11.16}$$

The above reasoning is very common, quite applicable in many situations. By the way, note that $\sum_{i,j} \pi_{(i,j)} j\sigma q = \sigma q E N$, where N is the number of members of group 2.

11.6 Birth/Death Processes

We noted earlier that the system of equations for the π_i may not be easy to solve. In many cases, for instance, the state space is infinite and thus the system of equations is infinite too. However, there is a rich class of Markov chains for which closed-form solutions have been found, called **birth/death processes**.[1]

Here the state space consists of (or has been mapped to) the set of nonnegative integers, and p_{ji} is nonzero only in cases in which $|i - j| = 1$. (The name "birth/death" has its origin in Markov models of biological populations, in which the state is the current population size.) Note for instance that the example of the gracefully degrading system above has this form. An M/M/1 queue—one server, "Markov" (i.e. exponential) interarrival times and Markov service times—is also a birth/death process, with the state being the number of jobs in the system.

Because the p_{ji} have such a simple structure, there is hope that we can find a closed-form solution for the π_i, and it turns out we can. Let $u_i = \rho_{i,i+1}$ and $d_i = \rho_{i,i-1}$ ('u' for "up," 'd' for "down"). Then (**??**) is

$$\pi_{i+1} d_{i+1} + \pi_{i-1} u_{i-1} = \pi_i \lambda_i = \pi_i (u_i + d_i),\ i \geq 1 \tag{11.17}$$

$$\pi_1 d_1 = \pi_0 \lambda_0 = \pi_0 u_0 \tag{11.18}$$

In other words,

$$\pi_{i+1} d_{i+1} - \pi_i u_i = \pi_i d_i - \pi_{i-1} u_{i-1},\ i \geq 1 \tag{11.19}$$

[1] Though we treat the continuous-time case here, there is also a discrete-time analog.

$$\pi_1 d_1 - \pi_0 u_0 = 0 \tag{11.20}$$

Applying (11.19) recursively to the base (11.20), we see that

$$\pi_i d_i - \pi_{i-1} u_{i-1} = 0,\ i \geq 1 \tag{11.21}$$

so that

$$\pi_i = \pi_{i-1} \frac{u_{i-1}}{d_i}\ i \geq 1 \tag{11.22}$$

and thus

$$\pi_i = \pi_0 r_i \tag{11.23}$$

where

$$r_i = \Pi_{k=1}^{i} \frac{u_{k-1}}{d_k} \tag{11.24}$$

where $r_i = 0$ for $i > m$ if the chain has no states past m.

Then since the π_i must sum to 1, we have that

$$\pi_0 = \frac{1}{1 + \sum_{i=1}^{\infty} r_i} \tag{11.25}$$

and the other π_i are then found via (11.23).

Note that the chain might be finite, i.e. have $u_i = 0$ for some i. In that case it is still a birth/death chain, and the formulas above for π still apply.

11.7 Cell Communications Model

Let's consider a more modern example of this sort, involving cellular phone systems. (This is an extension of the example treated in K.S. Trivedi, *Probability and Statistics, with Reliability and Computer Science Applications* (second edition), Wiley, 2002, Sec. 8.2.3.2, which is in turn is based on two papers in the *IEEE Transactions on Vehicular Technology*.)

We consider one particular cell in the system. Mobile phone users drift in and out of the cell as they move around the city. A call can either be a **new call**, i.e. a call which someone has just dialed, or a **handoff call**, i.e. a call which had already been in progress in a neighboring cell but now has moved to this cell.

Each call in a cell needs a **channel**.[2] There are n channels available in the cell. We wish to give handoff calls priority over new calls.[3] This is accomplished as follows.

The system always reserves g channels for handoff calls. When a request for a new call (i.e. a non-handoff call) arrives, the system looks at X_t, the current number of calls in the cell. If that number is less than n-g, so that there are more than g idle channels available, the new call is accepted; otherwise it is rejected.

We assume that new calls originate from within the cells according to a Poisson process with rate λ_1, while handoff calls drift in from neighboring cells at rate λ_2. Meanwhile, call durations are exponential with rate μ_1, while the time that a call remains within the cell is exponential with rate μ_2.

11.7.1 Stationary Distribution

We again have a birth/death process, though a bit more complicated than our earlier ones. Let $\lambda = \lambda_1 + \lambda_2$ and $\mu = \mu_1 + \mu_2$. Then here is a sample balance equation, focused on transitions into (left-hand side in the equation) and out of (right-hand side) state 1:

$$\pi_0\lambda + \pi_2 2\mu = \pi_1(\lambda + \mu) \tag{11.26}$$

Here's why: How can we enter state 1? Well, we could do so from state 0, where there are no calls; this occurs if we get a new call (rate λ_1) or a handoff call (rate λ_2. In state 2, we enter state 1 if one of the two calls ends (rate μ_1) or one of the two calls leaves the cell (rate μ_2). The same kind of reasoning shows that we leave state 1 at rate $\lambda + \mu$.

As another example, here is the equation for state n-g:

$$\pi_{n-g}[\lambda_2 + (n-g)\mu] = \pi_{n-g+1} \cdot (n-g+1)\mu + \pi_{n-g-1}\lambda \tag{11.27}$$

Note the term λ_2 in (13.44), rather than λ as in (13.43).

[2] This could be a certain frequency or a certain time slot position.

[3] We would rather give the caller of a new call a polite rejection message, e.g. "No lines available at this time, than suddenly terminate an existing conversation.

Using our birth/death formula for the π_i, we find that

$$\pi_k = \begin{cases} \pi_0 \frac{A^k}{k!}, & \text{k} \leq \text{n-g} \\ \pi_0 \frac{A^{n-g}}{k!} A_1^{k-(n-g)}, & \text{k} \geq \text{n-g} \end{cases} \tag{11.28}$$

where $A = \lambda/\mu$, $A_1 = \lambda_2/\mu$ and

$$\pi_0 = \left[\sum_{k=0}^{n-g-1} \frac{A^k}{k!} + \sum_{k=n-g}^{n} \frac{A^{n-g}}{k!} A_1^{k-(n-g)} \right]^{-1} \tag{11.29}$$

11.7.2 Going Beyond Finding the π

One can calculate a number of interesting quantities from the π_i:

- The probability of a handoff call being rejected is π_n.
- The probability of a new call being blocked is

$$\sum_{k=n-g}^{n} \pi_k \tag{11.30}$$

- Since the per-channel utilization in state i is i/n, the overall long-run per-channel utilization is

$$\sum_{i=0}^{n} \pi_i \frac{i}{n} \tag{11.31}$$

- The long-run proportion of accepted calls which are handoff calls is the rate at which handoff calls are accepted, divided by the rate at which calls are accepted:

$$\frac{\lambda_2 \sum_{i=0}^{n-1} \pi_i}{\lambda_1 \sum_{i=0}^{n-g-1} \pi_i + \lambda_2 \sum_{i=0}^{n-1} \pi_i} \tag{11.32}$$

Chapter 12

Advanced Markov Chains

One of the most famous stochastic models is that of a **Markov chain**. This type of model is widely used in computer science, biology, physics, business and so on.

(Note to the reader: An introduction to the subject matter of this chapter was presented in Chapter 6. The present chapter is mostly self-contained, but it may be helpful to read or review that previous chapter for additional examples.)

12.1 Discrete-Time Markov Chains

12.1.1 Example: Finite Random Walk

To motivate this discussion, let us start with a simple example: Consider a **random walk** on the set of integers between 1 and 5, moving randomly through that set, say one move per second, according to the following scheme. If we are currently at position i, then one time period later we will be at either i-1, i or i+1, according to the outcome of rolling a fair die—we move to i-1 if the die comes up 1 or 2, stay at i if the die comes up 3 or 4, and move to i+1 in the case of a 5 or 6. For the special cases i = 1 and i = 5, we simply move back to 2 or 4, respectively. (In random walk terminology, these are called **reflecting barriers**.)

The integers 1 through 5 form the **state space** for this process; if we are currently at 4, for instance, we say we are in state 4. Let X_t represent the position of the particle at time t, t = 0, 1,2,.... Typically X_0 is nonrandom.

The random walk is a **Markov process**. The process is "memoryless," meaning that we can "forget the past"; given the present and the past, the future depends only on the present. If the

X_i are discrete random variables, that means

$$P(X_{t+1} = s_{t+1}|X_t = s_t, X_{t-1} = s_{t-1}, \ldots, X_0 = s_0) = P(X_{t+1} = s_{t+1}|X_t = s_t) \tag{12.1}$$

In the continuous X_i case, replace $X_{t+1} = s_{t+1}$ above with $X_{t+1} \in (a, b)$. The term *Markov process* is the general one. If the state space is discrete, i.e. finite or countably infinite, then we usually use the more specialized term, *Markov chain*.[1]

Although this equation has a very complex look, it has a very simple meaning: The distribution of our next position, given our current position and all our past positions, is dependent only on the current position. In other words, the system is "memoryless," somewhat in analogy to the properties of the exponential distribution discussed in Section 9.2. (In fact exponential distributions will play a key role when we get to continuous-time Markov chains in Section **??**.) It is clear that the random walk process above does have this memoryless property; for instance, if we are now at position 4, the probability that our next state will be 3 is 1/3—no matter where we were in the past.

Continuing this example, let p_{ij} denote the probability of going from position i to position j in one step. For example, $p_{21} = p_{23} = \frac{1}{3}$ while $p_{24} = 0$ (we can reach position 4 from position 2 in two steps, but not in one step). The numbers p_{ij} are called the **one-step transition probabilities** of the process. Denote by P the matrix whose entries are the p_{ij}:

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 0 \\ 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 \\ 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{12.2}$$

By the way, it turns out (Section 6.1) that the matrix P^k gives the k-step transition probabilities. In other words, the element (i,j) of this matrix gives the probability of going from i to j in k steps.

12.1.2 Long-Run Distribution

In typical applications we are interested in the long-run distribution of the process, for example the long-run proportion of the time that we are at position 4. For each state i, define

$$\pi_i = \lim_{t \to \infty} \frac{N_{it}}{t} \tag{12.3}$$

[1] Be careful not to confuse the discrete or continuous nature of the state space with discrete or continuous time.

where N_{it} is the number of visits the process makes to state i among times 1, 2,..., t. In most practical cases, this proportion will exist and be independent of our initial position X_0. The π_i are called the **steady-state probabilities**, or the **stationary distribution** of the Markov chain.

Intuitively, the existence of π_i implies that as t approaches infinity, the system approaches steady-state, in the sense that

$$\lim_{t\to\infty} P(X_t = i) = \pi_i \tag{12.4}$$

Actually, the limit (12.4) may not exist in some cases. We'll return to that point later, but for typical cases it does exist, and we will usually assume this.

12.1.2.1 The Balance Equations

Equation (12.4) suggests a way to calculate the values π_i, as follows.

First note that

$$\begin{aligned} P(X_{t+1} = i) &= \sum_k P(X_t = k \text{ and } X_{t+1} = i) && (12.5)\\ &= \sum_k P(X_t = k)\ P(X_{t+1} = i | X_t = k) && (12.6)\\ &= \sum_k P(X_t = k)\ p_{ki} && (12.7) \end{aligned}$$

where the sum goes over all states k. For example, in our random walk example above, we would have

$$\begin{aligned} P(X_{t+1} = 3) &= \sum_{k=1}^{5} P(X_t = k \text{ and } X_{t+1} = 3) && (12.8)\\ &= \sum_{k=1}^{5} P(X_t = k)\ P(X_{t+1} = 3 | X_t = k) && (12.9)\\ &= \sum_{k=1}^{5} P(X_t = k)\ p_{k3} && (12.10) \end{aligned}$$

Then as $t \to \infty$ in Equation (12.7), intuitively we would have

$$\pi_i = \sum_k \pi_k p_{ki} \tag{12.11}$$

Remember, here we know the p_{ki} and want to find the π_i. Solving these **balance equations** equations (one for each i), gives us the π_i.

For the random walk problem above, for instance, the solution is $\pi = (\frac{1}{11}, \frac{3}{11}, \frac{3}{11}, \frac{3}{11}, \frac{1}{11})$. Thus in the long run we will spend 1/11 of our time at position 1, 3/11 of our time at position 2, and so on.

12.1.2.2 Solving the Balance Equations

A matrix formulation is also useful. Letting π denote the row vector of the elements π_i, i.e. $\pi = (\pi_1, \pi_2, ...)$, these equations (one for each i) then have the matrix form

$$\pi = \pi P \tag{12.12}$$

or

$$(I - P')\pi' = 0 \tag{12.13}$$

where as usual $'$ denotes matrix transpose.

Note that there is also the constraint

$$\sum_i \pi_i = 1 \tag{12.14}$$

One of the equations in the system is redundant. We thus eliminate one of them, say by removing the last row of I-P in (12.13). This can be used to calculate the π_i.

To reflect (12.14), which in matrix form is

$$1_n' \pi' = 1 \tag{12.15}$$

where 1_n is a column vector of all 1s, n is the number of states, and we replace the removed row in I-P by a row of all 1s, and in the right-hand side of (12.13) we replace the last 0 by a 1. We can then solve the system.

All this can be done with R's **solve()** function:

```
findpi1 <- function(p) {
   n <- nrow(p)
   imp <- diag(n) - t(p)  # I-P'
   imp[n,] <- rep(1,n)  # form row of 1s
   rhs <- c(rep(0,n-1),1)  # form right-hand-side vector
   pivec <- solve(imp,rhs)  # solve for pi
   return(pivec)
}
```

Or one can note from (12.12) that π is a left eigenvector of P with eigenvalue 1, so one can use R's **eigen()** function. It can be proven that if P is irreducible and aperiodic (defined later in this chapter), every eigenvalue other than 1 is smaller than 1 (so we can speak of *the* eigenvalue 1), and the eigenvector corresponding to 1 has all components real.

Since π is a left eigenvector, the argument in the call to **eigen()** must be P' rather than P. In addition, since an eigenvector is only unique up to scalar multiplication, we must deal with the fact that the return value of **eigen()** may have negative components, and will likely not satisfy (12.14). Here is the code:

```
findpi2 <- function(p) {
   n <- nrow(p)
   # find first eigenvector of P'
   pivec <- eigen(t(p))$vectors[,1]
   # guaranteed to be real, but could be negative
   if (pivec[1] < 0) pivec <- -pivec
   # normalize
   pivec <- pivec / sum(pivec)
   return(pivec)
}
```

But Equation (12.13) may not be easy to solve. For instance, if the state space is infinite, then this matrix equation represents infinitely many scalar equations. In such cases, you may need to try to find some clever trick which will allow you to solve the system, or in many cases a clever trick to analyze the process in some way other than explicit solution of the system of equations.

And even for finite state spaces, the matrix may be extremely large. In some cases, you may need to resort to numerical methods.

12.1.2.3 Periodic Chains

Note again that even if Equation (12.13) has a solution, this does not imply that (12.4) holds. For instance, suppose we alter the random walk example above so that

$$p_{i,i-1} = p_{i,i+1} = \frac{1}{2} \tag{12.16}$$

for i = 2, 3, 4, with transitions out of states 1 and 5 remaining as before. In this case, the solution to Equation (12.13) is $(\frac{1}{8}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{8})$. This solution is still valid, in the sense that Equation (12.3) will hold. For example, we will spend 1/4 of our time at Position 4 in the long run. But the limit of $P(X_i = 4)$ will not be 1/4, and in fact the limit will not even exist. If say X_0 is even, then X_i can be equal to 4 only for even values of i. We say that this Markov chain is **periodic** with period 2, meaning that returns to a given state can only occur after amounts of time which are multiples of 2.

12.1.2.4 The Meaning of the Term "Stationary Distribution"

In most analyses, X_0 is treated as constant, and typically not paid much attention. Our above limit theorems say, basically, "No matter what state we start in, the chain will converge to π." But we can model it as random too, which in this section will be quite useful

Though we have informally defined the term *stationary distribution* in terms of long-run proportions, the technical definition is this:

Definition 21 *Consider a Markov chain. Suppose we have a vector π of nonnegative numbers that sum to 1. Let X_0 have the distribution π. If that results in X_1 having that distribution too (and thus also all X_n), we say that π is the* **stationary distribution** *of this Markov chain.*

Note that this definition stems from (12.7).

For instance, in our (first) random walk example above, this would mean that if we have X_0 distributed on the integers 1 through 5 with probabilities $(\frac{1}{11}, \frac{3}{11}, \frac{3}{11}, \frac{3}{11}, \frac{1}{11})$, then for example $P(X_1 = 1) = \frac{1}{11}$, $P(X_1 = 4) = \frac{3}{11}$ etc. This is indeed the case, as you can verify using (12.7) with t = 0.

In our "notebook" view, here is what we would do. Imagine that we generate a random integer between 1 and 5 according to the probabilities $(\frac{1}{11}, \frac{3}{11}, \frac{3}{11}, \frac{3}{11}, \frac{1}{11})$,[2] and set X_0 to that number. We would then generate another random number, by rolling an ordinary die, and going left, right or

[2]Say by rolling an 11-sided die.

staying put, with probability 1/3 each. We would then write down X_1 and X_2 on the first line of our notebook. We would then do this experiment again, recording the results on the second line, then again and again. In the long run, 3/11 of the lines would have, for instance, $X_0 = 4$, and 3/11 of the lines would have $X_1 = 4$. In other words, X_1 would have the same distribution as X_0.

By the way, by making X_0 random, our Markov chain is no longer Markovian! This may seem counterintuitive, but by making X_0 we now have a mixture distribution. Recall the trick coin example in our mixture chapter, Chapter 24.1. There a sequence of independent random variables became dependent once the mixing variable (choice of coin) is introduced.

12.1.3 Example: Stuck-At 0 Fault

12.1.3.1 Description

In the above example, the labels for the states consisted of single integers i. In some other examples, convenient labels may be r-tuples, for example 2-tuples (i,j).

Consider a serial communication line. Let $B_1, B_2, B_3, ...$ denote the sequence of bits transmitted on this line. It is reasonable to assume the B_i to be independent, and that $P(B_i = 0)$ and $P(B_i = 1)$ are both equal to 0.5.

Suppose that the receiver will eventually fail, with the type of failure being **stuck at 0**, meaning that after failure it will report all future received bits to be 0, regardless of their true value. Once failed, the receiver stays failed, and should be replaced. Eventually the new receiver will also fail, and we will replace it; we continue this process indefinitely.

Let ρ denote the probability that the receiver fails on any given bit, with independence between bits in terms of receiver failure. Then the lifetime of the receiver, that is, the time to failure, is geometrically distributed with "success" probability ρ i.e. the probability of failing on receipt of the i-th bit after the receiver is installed is $(1 - \rho)^{i-1}\rho$ for i = l,2,3,...

However, the problem is that we will not know whether a receiver has failed (unless we test it once in a while, which we are not including in this example). If the receiver reports a long string of 0s, we should suspect that the receiver has failed, but of course we cannot be sure that it has; it is still possible that the message being transmitted just happened to contain a long string of 0s.

Suppose we adopt the policy that, if we receive k consecutive 0s, we will replace the receiver with a new unit. Here k is a design parameter; what value should we choose for it? If we use a very small value, then we will incur great expense, due to the fact that we will be replacing receiver units at an unnecessarily high rate. On the other hand, if we make k too large, then we will often wait too long to replace the receiver, and the resulting error rate in received bits will be sizable. Resolution of this tradeoff between expense and accuracy depends on the relative importance of the two. (There

are also other possibilities, involving the addition of redundant bits for error detection, such as parity bits. For simplicity, we will not consider such refinements here. However, the analysis of more complex systems would be similar to the one below.)

12.1.3.2 Initial Analysis

A natural state space in this example would be

$$\{(i,j) : i = 0, 1, ..., k-1; j = 0, 1; i + j \neq 0\} \tag{12.17}$$

where i represents the number of consecutive 0s that we have received so far, and j represents the state of the receiver (0 for failed, 1 for nonfailed). Note that when we are in a state of the form (k-1,j), if we receive a 0 on the next bit (whether it is a true 0 or the receiver has failed), our new state will be (0,1), as we will install a new receiver. Note too that there is no state (0,0), since if the receiver is down it must have received at least one bit.

The calculation of the transition matrix P is straightforward, though it requires careful thought. For example, suppose the current state is (2,1), and that we are investigating the expense and bit accuracy corresponding to a policy having k = 5. What can happen upon receipt of the next bit? The next bit will have a true value of either 0 or 1, with probability 0.5 each. The receiver will change from working to failed status with probability ρ. Thus our next state could be:

- (3,1), if a 0 arrives, and the receiver does not fail;
- (0,1), if a 1 arrives, and the receiver does not fail; or
- (3,0), if the receiver fails

The probabilities of these three transitions out of state (2,1) are:

$$p_{(2,1),(3,1)} = 0.5(1-\rho) \tag{12.18}$$
$$p_{(2,1),(0,1)} = 0.5(1-\rho) \tag{12.19}$$
$$p_{(2,1),(3,0)} = \rho \tag{12.20}$$

Other entries of the matrix P can be computed similarly. Note by the way that from state (4,1) we will go to (0,1), no matter what happens.

Formally specifying the matrix P using the 2-tuple notation as above would be very cumbersome. In this case, it would be much easier to map to a one-dimensional labeling. For example, if k =

5, the nine states (1,0),...,(4,0),(0,1),(1,1),...,(4,1) could be renamed states 1,2,...,9. Then we could form P under this labeling, and the transition probabilities above would appear as

$$p_{78} = 0.5(1-\rho) \tag{12.21}$$
$$p_{75} = 0.5(1-\rho) \tag{12.22}$$
$$p_{73} = \rho \tag{12.23}$$

12.1.3.3 Going Beyond Finding π

Finding the π_i should be just the first step. We then want to use them to calculate various quantities of interest.[3] For instance, in this example, it would also be useful to find the error rate ϵ, and the mean time (i.e., the mean number of bit receptions) between receiver replacements, μ. We can find both ϵ and μ in terms of the π_i, in the following manner.

The quantity ϵ is the proportion of the time during which the true value of the received bit is 1 but the receiver is down, which is 0.5 times the proportion of the time spent in states of the form (i,0):

$$\epsilon = 0.5(\pi_1 + \pi_2 + \pi_3 + \pi_4) \tag{12.24}$$

This should be clear intuitively, but it would also be instructive to present a more formal derivation of the same thing. Let E_n be the event that the n-th bit is received in error, with D_n denoting the event that the receiver is down. Then

$$\epsilon = \lim_{n\to\infty} P(E_n) \tag{12.25}$$
$$= \lim_{n\to\infty} P(B_n = 1 \text{ and } D_n) \tag{12.26}$$
$$= \lim_{n\to\infty} P(B_n = 1)P(D_n) \tag{12.27}$$
$$= 0.5(\pi_1 + \pi_2 + \pi_3 + \pi_4) \tag{12.28}$$

Here we used the fact that B_n and the receiver state are independent.

Note that with the interpretation of π as the stationary distribution of the process, in Equations (12.25) above, we do not even need to take limits.

[3] Note that unlike a classroom setting, where those quantities would be listed for the students to calculate, in research we must decide on our own which quantities are of interest.

Equations (12.25) follow a pattern we'll use repeatedly in this chapter. In subsequent examples we will not show the steps with the limits, but the limits are indeed there. Make sure to mentally go through these steps yourself.[4]

Now to get μ in terms of the π_i note that since μ is the long-run average number of bits between receiver replacements, it is then the reciprocal of η, the long-run fraction of bits that result in replacements. For example, say we replace the receiver on average every 20 bits. Over a period of 1000 bits, then (speaking on an intuitive level) that would mean about 50 replacements. Thus approximately 0.05 (50 out of 1000) of all bits result in replacements. In other words,

$$\mu = \frac{1}{\eta} \tag{12.29}$$

Again suppose k = 5. A replacement will occur only from states of the form (4,j), and even then only under the condition that the next reported bit is a 0. In other words, there are three possible ways in which replacement can occur:

(a) We are in state (4,0). Here, since the receiver has failed, the next reported bit will definitely be a 0, regardless of that bit's true value. We will then have a total of k = 5 consecutive received 0s, and therefore will replace the receiver.

(b) We are in the state (4,1), and the next bit to arrive is a true 0. It then will be reported as a 0, our fifth consecutive 0, and we will replace the receiver, as in (a).

(c) We are in the state (4,1), and the next bit to arrive is a true 1, but the receiver fails at that time, resulting in the reported value being a 0. Again we have five consecutive reported 0s, so we replace the receiver.

Therefore,

$$\eta = \pi_4 + \pi_9(0.5 + 0.5\rho) \tag{12.30}$$

Again, make sure you work through the full version of (12.30), using the pattern in (12.25).

Thus

$$\mu = \frac{1}{\eta} = \frac{1}{\pi_4 + 0.5\pi_9(1+\rho)} \tag{12.31}$$

[4]The other way to work this out rigorously is to assume that X_0 has the distribution π, as in Section 12.1.2.4. Then no limits are needed in (12.25). But this may be more difficult to understand.

This kind of analysis could be used as the core of a cost-benefit tradeoff investigation to determine a good value of k. (Note that the π_i are functions of k, and that the above equations for the case k = 5 must be modified for other values of k.)

12.1.4 Example: Shared-Memory Multiprocessor

(Adapted from *Probabiility and Statistics, with Reliability, Queuing and Computer Science Applicatiions*, by K.S. Trivedi, Prentice-Hall, 1982 and 2002, but similar to many models in the research literature.)

12.1.4.1 The Model

Consider a shared-memory multiprocessor system with m memory modules and m CPUs. The address space is partitioned into m chunks, based on either the most-significant or least-significant $\log_2 m$ bits in the address.[5]

The CPUs will need to access the memory modules in some random way, depending on the programs they are running. To make this idea concrete, consider the Intel assembly language instruction

```
add %eax, (%ebx)
```

which adds the contents of the EAX register to the word in memory pointed to by the EBX register. Execution of that instruction will (absent cache and other similar effects, as we will assume here and below) involve two accesses to memory—one to fetch the old value of the word pointed to by EBX, and another to store the new value. Moreover, the instruction itself must be fetched from memory. So, altogether the processing of this instruction involves three memory accesses.

Since different programs are made up of different instructions, use different register values and so on, the sequence of addresses in memory that are generated by CPUs are modeled as random variables. In our model here, the CPUs are assumed to act independently of each other, and successive requests from a given CPU are independent of each other too. A CPU will choose the i^{th} module with probability q_i. A memory request takes one unit of time to process, though the wait may be longer due to queuing. In this very simplistic model, as soon as a CPU's memory request is fulfilled, it generates another one. On the other hand, while a CPU has one memory request pending, it does not generate another.

Let's assume a crossbar interconnect, which means there are m^2 separate paths from CPUs to memory modules, so that if the m CPUs have memory requests to m different memory modules,

[5] You may recognize this as high-order and low-order interleaving, respectively.

then all the requests can be fulfilled simultaneously. Also, assume as an approximation that we can ignore communication delays.

How good are these assumptions? One weakness, for instance, is that many instructions, for example, do not use memory at all, except for the instruction fetch, and as mentioned, even the latter may be suppressed due to cache effects.

Another example of potential problems with the assumptions involves the fact that many programs will have code like

```
for (i = 0; i < 10000; i++) sum += x[i];
```

Since the elements of the array x will be stored in consecutive addresses, successive memory requests from the CPU while executing this code will not be independent. The assumption would be more justified if we were including cache effects, or if we are studying a timesharing system with a small quantum size.

Thus, many models of systems like this have been quite complex, in order to capture the effects of various things like caching, nonindependence and so on in the model. Nevertheless, one can often get some insight from even very simple models too. In any case, for our purposes here it is best to stick to simple models, so as to understand more easily.

Our state will be an m-tuple $(N_1, ..., N_m)$, where N_i is the number of requests currently pending at memory module i. Recalling our assumption that a CPU generates another memory request immediately after the previous one is fulfilled, we always have that $N_1 + ... + N_m = m$.

It is straightforward to find the transition probabilities p_{ij}. Here are a couple of examples, with m = 2:

- $p_{(2,0),(1,1)}$: Recall that state (2,0) means that currently there are two requests pending at Module 1, one being served and one in the queue, and no requests at Module 2. For the transition $(2,0) \to (1,1)$ to occur, when the request being served at Module 1 is done, it will make a new request, this time for Module 2. This will occur with probability q_2. Meanwhile, the request which had been queued at Module 1 will now start service. So, $p_{(2,0),(1,1)} = q_2$.

- $p_{(1,1),(1,1)}$: In state (1,1), both pending requests will finish in this cycle. To go to (1,1) again, that would mean that the two CPUs request different modules from each other—CPUs 1 and 2 choose Modules 1 and 2 or 2 and 1. Each of those two possibilities has probability $q_1 q_2$, so $p_{(1,1),(1,1)} = 2q_1 q_2$.

We then solve for the π, using (12.11). It turns out, for example, that

$$\pi_{(1,1)} = \frac{q_1 q_2}{1 - 2q_1 q_2} \tag{12.32}$$

12.1.4.2 Going Beyond Finding π

Let B denote the number of memory requests completed in a given memory cycle. Then we may be interested in E(B), the number of requests completed per unit time, i.e. per cycle. We can find E(B) as follows. Let S denote the current state. Then, continuing the case m = 2, we have from the Law of Total Expectation (Section 24.4),[6]

$$\begin{aligned} E(B) &= E[E(B|S)] && (12.33)\\ &= P(S = (2,0))E(B|S = (2,0)) + P(S = (1,1))E(B|S = (1,1)) + P(S = (0,2))E(B|S = (0,2)) && (12.34)\\ &= \pi_{(2,0)}E(B|S = (2,0)) + \pi_{(1,1)}E(B|S = (1,1)) + \pi_{(0,2)}E(B|S = (0,2)) && (12.35) \end{aligned}$$

All this equation is doing is finding the overall mean of B by breaking down into the cases for the different states.

Now if we are in state (2,0), only one request will be completed this cycle, and B will be 1. Thus $E(B|S = (2,0)) = 1$. Similarly, $E(B|S = (1,1)) = 2$ and so on. After doing all the algebra, we find that

$$EB = \frac{1 - q_1 q_2}{1 - 2q_1 q_2} \tag{12.36}$$

The maximum value of E(B) occurs when $q_1 = q_2 = \frac{1}{2}$, in which case E(B)=1.5. This is a lot less than the maximum capacity of the memory system, which is m = 2 requests per cycle.

So, we can learn a lot even from this simple model, in this case learning that there may be a substantial underutilization of the system. This is a common theme in probabilistic modeling: Simple models may be worthwhile in terms of insight provided, even if their numerical predictions may not be too accurate.

[6] Actually, we could take a more direct route in this case, noting that B can only take on the values 1 and 2. Then $EB = P(B = 1) + 2P(B = 2) = \pi_{(2,0} + \pi_{s(0,2)} + 2\pi_{(1,1)}$. But the analysis below extends better to the case of general m.

12.1.5 Example: Slotted ALOHA

Recall the slotted ALOHA model from Chapter 2:

- Time is divided into slots or epochs.
- There are n nodes, each of which is either idle or has a **single** message transmission pending. So, a node doesn't generate a new message until the old one is successfully transmitted (a very unrealistic assumption, but we're keeping things simple here).
- In the middle of each time slot, each of the idle nodes generates a message with probability q.
- Just before the end of each time slot, each active node attempts to send its message with probability p.
- If more than one node attempts to send within a given time slot, there is a **collision**, and each of the transmissions involved will fail.
- So, we include a **backoff** mechanism: At the middle of each time slot, each node with a message will with probability q attempt to send the message, with the transmission time occupying the remainder of the slot.

So, q is a design parameter, which must be chosen carefully. If q is too large, we will have too mnay collisions, thus increasing the average time to send a message. If q is too small, a node will often refrain from sending even if no other node is there to collide with.

Define our state for any given time slot to be the number of nodes currently having a message to send at the very beginning of the time slot (before new messages are generated). Then for $0 < i < n$ and $0 < j < n - i$ (there will be a few special boundary cases to consider too), we have

$$p_{i,i-1} = \underbrace{(1-q)^{n-i}}_{\text{no new msgs}} \cdot \underbrace{i(1-p)^{i-1}p}_{\text{one xmit}} \tag{12.37}$$

$$p_{ii} = \underbrace{(1-q)^{n-i} \cdot [1 - i(1-p)^{i-1}p]}_{\text{no new msgs and no succ xmits}} + \underbrace{(n-i)(1-q)^{n-i-1}q \cdot (i+1)(1-p)^{i}p}_{\text{one new msg and one xmit}} \tag{12.38}$$

$$
\begin{aligned}
p_{i,i+j} &= \underbrace{\binom{n-i}{j} q^j (1-q)^{n-i-j} \cdot [1-(i+j)(1-p)^{i+j-1}p]}_{\text{j new msgs and no succ xmit}} \\
&+ \underbrace{\binom{n-i}{j+1} q^{j+1} (1-q)^{n-i-j-1} \cdot (i+j+1)(1-p)^{i+j}p}_{\text{j+1 new msgs and succ xmit}}
\end{aligned}
\tag{12.39}
$$

Note that in (12.38) and (12.39), we must take into account the fact that a node with a newly-created messages might try to send it. In (12.39), for instance, in the first term we have j new messages, on top of the i we already had, so i+j messages might try to send. The probability that there is no successful transmission is then $1-(i+j)(1-p)^{i+j-1}p$.

The matrix P is then quite complex. We always hope to find a closed-form solution, but that is unlikely in this case. Solving it on a computer is easy, though, say by using the **solve()** function in the R statistical language.

12.1.5.1 Going Beyond Finding π

Once again various interesting quantities can be derived as functions of the π, such as the system throughput τ, i.e. the number of successful transmissions in the network per unit time. Here's how to get τ:

First, suppose for concreteness that in steady-state the probability of there being a successful transmission in a given slot is 20%. Then after, say, 100,000 slots, about 20,000 will have successful transmissions—a throughput of 0.2. So, the long-run probability of successful transmission is the same as the long-run fraction of slots in which there are successful transmissions! That in turn can be broken down in terms of the various states:

$$
\begin{aligned}
\tau &= P(\text{success xmit}) \\
&= \sum_s P(\text{success xmit} \mid \text{in state s}) P(\text{in state s})
\end{aligned}
\tag{12.40}
$$

Now, to calculate P(success xmit | in state s), recall that in state s we start the slot with s nonidle nodes, but that we may acquire some new ones; each of the n-s idle nodes will create a new message,

with probability q. So,

$$P(\text{success xmit} \mid \text{in state s}) = \sum_{j=0}^{n-s} \binom{n-s}{j} q^j (1-q)^{n-s-j} \cdot (s+j)(1-p)^{s+j-1} p \tag{12.41}$$

Substituting into (12.40), we have

$$\tau = \sum_{s=0}^{n} \sum_{j=0}^{n-s} \binom{n-s}{j} q^j (1-q)^{n-s-j} \cdot (s+j)(1-p)^{s+j-1} p \cdot \pi_s \tag{12.42}$$

With some more subtle reasoning, one can derive the mean time a message waits before being successfully transmitted, as follows:

Focus attention on one particular node, say Node 0. It will repeatedly cycle through idle and busy periods, I and B. We wish to find E(B). I has a geometric distribution with parameter q,[7] so

$$E(I) = \frac{1}{q} \tag{12.43}$$

Then if we can find E(I+B), we will get E(B) by subtraction.

To find E(I+B), note that there is a one-to-one correspondence between I+B cycles and successful transmissions; each I+B period ends with a successful transmission at Node 0. Imagine again observing this node for, say, 100000 time slots, and say E(I+B) is 2000. That would mean we'd have about 50 cycles, thus 50 successful transmissions from this node. In other words, the throughput would be approximately 50/100000 = 0.02 = 1/E(I+B). So, a fraction

$$\frac{1}{E(I+B)} \tag{12.44}$$

of the time slots have successful transmissions from this node.

But that quantity is the throughput for this node (number of successful transmissions per unit time), and due to the symmetry of the system, that throughput is 1/n of the total throughput of the n nodes in the network, which we denoted above by τ.

[7] If a message is sent in the same slot in which it is created, we will count B as 1. If it is sent in the following slot, B = 2, etc. B will have a modified geometric distribution starting at 0 instead of 1, but we will ignore this here for the sake of simplicity.

So,

$$E(I+B) = \frac{n}{\tau} \tag{12.45}$$

Thus from (12.43) we have

$$E(B) = \frac{n}{\tau} - \frac{1}{q} \tag{12.46}$$

where of course τ is the function of the π_i in (12.40).

Now let's find the proportion of attempted transmissions which are successful. This will be

$$\frac{E(\text{number of successful transmissions in a slot})}{E(\text{number of attempted transmissions in a slot})} \tag{12.47}$$

(To see why this is the case, again think of watching the network for 100,000 slots.) Then the proportion of successful transmissions during that period of time is the number of successful transmissions divided by the number of attempted transmissions. Those two numbers are approximately the numerator and denominator of 12.47.

Now, how do we evaluate (12.47)? Well, the numerator is easy, since it is τ, which we found before. The denominator will be

$$\sum_s \pi_s [sp + (n-s)pq] \tag{12.48}$$

The factor sp+spq comes from the following reasoning. If we are in state s, the s nodes which already have something to send will each transmit with probability p, so there will be an expected number sp of them that try to send. Also, of the n-s which are idle at the beginning of the slot, an expected sq of them will generate new messages, and of those sq, and estimated sqp will try to send.

12.2 Simulation of Markov Chains

Simulation of Markov chains is identical to the patterns we've seen in earlier chapters, except for one somewhat subtle difference. To see this, consider the first simulation code presented in this book, in Section 2.14.6.

There we were simulating X_1 and X_2, the state of the system during the first two time slots. A rough outline of the code is

```
do nreps times
   simulate X1 and X2
   record X1, X1 and update counts
calculate probabilities as counts/nreps
```

We "played the movie" **nreps** times, calculating the behavior of X_1 and X_2 over many plays.

But suppose instead that we had been interested in finding

$$\lim_{n \to \infty} \frac{X_1 + ... + X_n}{n} \tag{12.49}$$

i.e. the long-run average number of active nodes over infinitely many time slots. **In that case, we would need to play the movie only once.**

Here's an example, simulating the stuck-at 0 example from Section 12.1.3:

```
1  # simulates the stuck-at 0 fault example, finding mean time between
2  # replacements; we'll keep simulating until we have nreplace replacements
3  # of the receiver, then divide that into the number of bits received, to
4  # get the mean time between replacements
5  sasim <- function(nreplace,rho,k) {
6     replace <- 0  # number of receivers replaced so far
7     up <- TRUE  # receiver is up
8     nbits <- 0  # number of bits received so far
9     ncsec0 <- 0  # current number of consecutive 0s
10    while (TRUE) {
11       bit <- sample(0:1,1)
12       nbits <- nbits + 1
13       if (runif(1) < rho) {
14          up <- FALSE
15          bit <- 0
16       }
17       if (bit == 0) {
18          ncsec0 <- ncsec0 + 1
19          if (ncsec0 == k) {
20             replace <- replace + 1
21             ncsec0 <- 0
22             up <- TRUE
23          }
24       }
25       if (replace == nreplace) break
26    }
27    return(nbits/nreplace)
28 }
```

This follows from the fact that the limit in (12.3) occurs even in "one play."

12.3 Some Mathematical Conditions

There is a rich mathematical theory regarding the asymptotic behavior of Markov chains. We will not present such material here in this brief introduction, but we will give an example of the implications the theory can have.

(Note: Due to the large number of definitions and properties in this section, they are highlighted via bulleting.)

- A state in a Markov chain is called **recurrent** if it has the property that if we start at that state, we are guaranteed to return to the state.
- A nonrecurrent state is called **transient.**

Let T_{ii} denote the time needed to return to state i if we start there. Keep in mind that T_{ii} is the time from one entry to state i to the next entry to state i. So, it includes time spent in i, which is 1 unit of time for a discrete-time chain and a random exponential amount of time in the continuous-time case, and then time spent away from i, up to the time of next entry to i. Note that an equivalent definition of recurrence is that $P(T_{ii} < \infty) = 1$, i.e. we are sure to return to i. By the Markov property, if we are sure to return once, then we are sure to return again once after that, and so on, so this implies infinitely many visits.

- A recurrent state i is called **positive recurrent** if $E(T_{ii}) < \infty$.[8]
- A state which is recurrent but not positive recurrent is called **null recurrent**.
- In the discrete time case, a state i is recurrent if and only if

$$\sum_{n=1}^{\infty} p_{ii}^{(n)} = \infty \tag{12.50}$$

  where $p_{uv}^{(k)}$ means the row u, column v element of the kth power of the transition matrix of the chain, i.e. the probability of being at state v after n steps if one starts at u.

This last can be easily seen in the "only if" case: Let A_n denote the indicator random variable for the event $T_{ii} = n$ (Section 3.9). Then

$$p_{ii}^{(n)} = EA_n, \tag{12.51}$$

[8]Some random variables are finite with probability 1 but have infinite expected value. For instance, suppose M = 1,2,3,... with probabilities 1/2, 1/4, 1/8... Those probabilities sum to 1, so $P(M < \infty) = 1$ yet $EM = \infty$.

so the left-hand side of (12.50) is

$$E\left(\sum_{n=1}^{\infty} A_n\right) \tag{12.52}$$

i.e. the expected value of the total number of visits to state i. If state i is recurrent, then we will visit i infinitely often, and thus that sum should be equal to infinity.

- A chain is **irreducible** if we can get from any state to any other state (though not necessarily in one step).[9]
- One can show that in an irreducible chain, if one state is recurrent then they all are. The same statement holds if "recurrent" is replaced by "positive recurrent."

Again, this last bullet should make intuitive sense to you for the recurrent case: We make infinitely many visits to state i, and each time we have a nonzero probability of going to state j from there. Thus eventually we will visit j, and indeed each time we visit i we will subsequently visit j at some time. Thus we will make infinitely many visits to j as well, i.e. j is recurrent.

12.3.1 Example: Random Walks

Consider the famous **random walk** on the full set of integers: At each time step, one goes left one integer or right one integer (e.g. to +3 or +5 from +4), with probability 1/2 each. In other words, we flip a coin and go left for heads, right for tails.

If we start at 0, then we return to 0 when we have accumulated an equal number of heads and tails. So for even-numbered n, i.e. n = 2m, we have

$$p_{ii}^{(n)} = P(\text{m heads and m tails}) = \binom{2m}{m}\frac{1}{2^{2m}} \tag{12.53}$$

One can use Stirling's approximation,

$$m! \approx \sqrt{2\pi}e^{-m}m^{m+1/2} \tag{12.54}$$

to show that the series (12.50) diverges in this case. So, this chain (meaning all states in the chain) is recurrent. However, it turns out not to be not positive recurrent, as we'll see below.

[9] Some readers may see the similarity to the notion of a connected graph.

The same is true for the corresponding random walk on the two-dimensional integer lattice (moving up, down, left or right with probability 1/4 each). However, in the three-dimensional and higher cases, the chain is not even null recurrent; it is transient.

12.3.2 Finding Hitting and Recurrence Times

We will use the symbol T_{ij} to denote the time it takes to get to state j if we are now in i, known as the **hitting time**, analogous to the **recurrence times** T_{ii}. Note that this is measured from the time that we enter state i to the time we enter state j.

12.3.3 Recurrence Times and Stationary Probabiilities

Consider a continuous-time chain. For a positive recurrent state i, it turns out that

$$\pi_i = \frac{1/\lambda_i}{E(T_{ii})} \tag{12.55}$$

To see this, we take an approach similar to that of Section 12.1.5.1. Define alternating On and Off subcycles, where On means we are at state i and Off means we are elsewhere. Define a full cycle to consist of an On subcycle followed by an Off subcycle. Note again that T_{ii} is measured from the time we enter state i once until the time we enter it again.

Then intuitively the long-run proportion of time we are in state i is

$$\pi_i = \frac{E(\text{On})}{E(T_{ii})} \tag{12.56}$$

But an On subcycle has an exponential distribution, with mean duration $1/\lambda_i$. That gives us (12.55).

In the discrete-time case, an On subcycle always has duration 1. Thus we have

$$\pi_i = \frac{1}{E(T_{ii})} \tag{12.57}$$

Thus positive recurrence means that $\pi_i > 0$. For a null recurrent chain, the limits in Equation (12.3) are 0, which means that there may be rather little one can say of interest regarding the long-run behavior of the chain.

Nevertheless, note that (12.57) makes sense even in the null recurrent case. The right-hand side will be 0, and if one views π_i in terms of (12.3), that means that in the long-run we don't spend a positive proportion of time at this state.

Equation (12.57) also shows that random walk in one or two dimensions, though recurrent, is null recurrent. If they were positive recurrent, then by symmetry all the π_i would have to be equal, and yet they would have to sum to 1, a contradiction. So, $ET_{ii} = \infty$ for all i.

12.3.3.1 Hitting Times

We are often interested in finding quantities of the form $E(T_{ij})$. We can do so by setting up systems of equations similar to the balance equations used for finding stationary distributions.

First consider the discrete case. Conditioning on the first step we take after being at state i, and using the Law of Total Expectation, we have

$$E(T_{ij}) = \sum_k p_{ik} E(T_{ij} \mid W = k) \tag{12.58}$$

where W is the first state we go to after leaving state i. There are two cases for $E(T_{ij} \mid W = k)$:

- **W = j:**

  This is the trivial case; $E(T_{ij} \mid W = k) = 1$.

- **W $\neq$ j:**

  If our first step is to some state k other than j, it takes us 1 unit of time to get to k, at which point "time starts over" (Markov property), and our expected remaining time is ET_{kj}. So, in this case $E(T_{ij} \mid W = k) = 1 + ET_{kj}$.

So, using the Law of Total Expectation, we have

$$E(T_{ij}) = \sum_k p_{ik} E(T_{ij} \mid W = k) = \sum_{k \neq j} p_{ik}[1 + E(T_{kj})] + p_{ij} \cdot 1 = 1 + \sum_{k \neq j} p_{ik} E(T_{kj}) \tag{12.59}$$

By varying i and j in (12.59), we get a system of linear equations which we can solve to find the ET_{ij}. Note that (12.57) gives us equations we can use here too.

The continuous version uses the same reasoning:

$$E(T_{ij}) = \sum_{k \neq j} p_{ik} \left[\frac{1}{\lambda_i} + E(T_{kj}) \right] + p_{ij} \cdot \frac{1}{\lambda_i} = \frac{1}{\lambda_i} + \sum_{k \neq j} p_{ik} E(T_{kj}) \tag{12.60}$$

One can use a similar analysis to determine the probability of ever reaching a state, in chains in which this probability is not 1. (Some chains have have transient or even **absorbing** states, i.e. states u such that $p_{uv} = 0$ whenever $v \neq u$.)

For fixed j define

$$\alpha_{ij} = P(T_{ij} < \infty) \tag{12.61}$$

Then denoting by S the state we next visit after i, we have

$$\begin{aligned}
\alpha_{ij} &= P(T_{ij} < \infty) && (12.62)\\
&= \sum_k P(S = k \text{ and } T_{ij} < \infty) && (12.63)\\
&= \sum_{k \neq j} P(S = k \text{ and } T_{kj} < \infty) + P(S = j) && (12.64)\\
&= \sum_{k \neq j} P(S = k)\ P(T_{kj} < \infty | S = k) + P(S = j) && (12.65)\\
&= \sum_{k \neq j} p_{ik}\ \alpha_{kj} + p_{ij} && (12.66)\\
& && (12.67)
\end{aligned}$$

So, again we have a system of linear equations that we can solve for the α_{ij}.

12.3.4 Example: Finite Random Walk

Let's go back to the example in Section 12.1.1.

Suppose we start our random walk at 2. How long will it take to reach state 4? Set $b_i = E(T_{i4}|\text{start at i})$. From (12.59) we could set up equations like

$$b_2 = \frac{1}{3}(1 + b_1) + \frac{1}{3}(1 + b_2) + \frac{1}{3}(1 + b_3) \tag{12.68}$$

Now change the model a little, and make states 1 and 6 absorbing. Suppose we start at position 3. What is the probability that we eventually are absorbed at 6 rather than 1? We could set up equations like (12.62) to find this.

12.3.5 Example: Tree-Searching

Consider the following Markov chain with infinite state space {0,1,2,3,...}.[10] The transition matrix is defined by $p_{i,i+1} = q_i$ and $p_{i0} = 1 - q_i$. This kind of model has many different applications, including in computer science tree-searching algorithms. (The state represents the level in the tree where the search is currently, and a return to 0 represents a backtrack. More general backtracking can be modeled similarly.)

The question at hand is, What conditions on the q_i will give us a positive recurrent chain?

Assuming $0 < q_i < 1$ for all i, the chain is clearly irreducible. Thus, to check for recurrence, we need check only one state, say state 0.

For state 0 (and thus the entire chain) to be recurrent, we need to show that $P(T_{00} < \infty) = 1$. But

$$P(T_{00} > n) = \Pi_{i=0}^{n-1} q_i \tag{12.69}$$

(In the case n = 0, define the product to be 1.)

Therefore, the chain is recurrent if and only if

$$\lim_{n \to \infty} \Pi_{i=0}^{n-1} q_i = 0 \tag{12.70}$$

For positive recurrence, we need $E(T_{00}) < \infty$. Now, one can show[11] that for any nonnegative integer-valued random variable Y

$$E(Y) = \sum_{n=0}^{\infty} P(Y > n) \tag{12.71}$$

[10] Adapted from *Performance Modelling of Communication Networks and Computer Architectures*, by P. Harrison and N. Patel, pub. by Addison-Wesley, 1993.

[11] In (3.13), replace c by $\sum_{i=1}^{c} 1$, yielding a double sum. Then reverse the order of summation. In the continuous case, the relation is $EY = \int_0^\infty P(Y > t)\ dt$.

Thus for positive recurrence, our condition on the q_i is

$$\sum_{n=0}^{\infty} \Pi_{i=0}^{n-1} q_i < \infty \tag{12.72}$$

12.4 Higher-Order Markov Chains

Recall that the Markov property can be summarized as:

The future depends only on the present, not the past.

This is stated formally as

$$P(X_{t+1} = s_{t+1} | X_t = s_t, X_{t-1} = s_{t-1}, \ldots, X_0 = s_0) = P(X_{t+1} = s_{t+1} | X_t = s_t) \tag{12.73}$$

But what if we extend this a bit to

The future depends only on the present, and the most recent past, not the full past.

i.e.

$$P(X_{t+1} = s_{t+1} | X_t = s_t, X_{t-1} = s_{t-1}, \ldots, X_0 = s_0) = P(X_{t+1} = s_{t+1} | X_t = s_t, X_{t-1} = s_{t-1}) \tag{12.74}$$

Denote our original, ordinary Markov chain, state space by {1,2,...,r}. The new space has r^2 elements, all possible pairs (i,j), with j meaning the present state in terms of the original matrix and i being the immediately previous one. The transition probabilities are now of the form

$$p_{(i,j),(j,k)} \tag{12.75}$$

It is now harder to decide how to model those probabilities, but in principle is can be done.

We can also construct third-order models, and so on.

12.5 Hidden Markov Models

Czech is article-free language"—a Czech colleague

The word *hidden* in the term *Hidden Markov Model* (HMM) refers to the fact that the state of the process is hidden, i.e. unobservable.

An HMM consists of a Markov chain X_n which is unobservable, together with observable values Y_n. The X_n are governed by the transition probabilities p_{ij}, and the Y_n are generated from the X_n according to

$$r_{ks} = P(Y_n = s|X_n = k) \tag{12.76}$$

Typically there is also a conditioning variable M, with the transition probability for the X_n being dependent on M. So, let's we write them as $p_{ij(m)}$. For each m, there is a different set of transition probabilities. Note that M itself has a distribution, with probabilities denoted q_k. M is unobserved.

A good example of HMMs would be in text mining applications, say in document classification. We may have different types of newspaper articles which we wish to automatically sort by category—sports, politics, etc.

Here the Y_n might be words in the text, and X_n would be their parts of speech (POS)—nouns, verbs, adjectives and so on. Consider the word *round*, for instance. Your first thought might be that it is an adjective, but it could be a noun (e.g. an elimination round in a tournament) or a verb (e.g. to round off a number or round a corner). "Time" would be the order of a word in the document; the first word would have time 1, the second word time 2, and so on. So, we'd like to have good guesses as to which usage of *round* we have, and so on. This is called **POS tagging**. Knowing the POS of the words in our document might help with further analysis, such as document classification.

M would be the document type.

Note that we have a mixture model here (Chapter 24), with M being the mixing variable.

Based on past data, we would know the probability $p_{ij(m)}$, r_{km} and q_k. The goal is generally to guess the X_n from the Y_n. The full details are too complex to give here, but really it is just one giant application of Bayes' Rule.

HMM models are also used in speech processing, DNA modeling and many other applications.

R's CRAN repository has various functions for HMM analysis, such as the **HiddenMarkov** library.

12.6 Further Reading

The classic book on Markov chains and other stochastic processes is *An Introduction to Stochastic Modeling*, Third Edition. S. Karlin and H. M. Taylor. Academic Press, 1998.

Exercises

1. Consider a "wraparound" variant of the random walk in Section 12.1.1. We still have a reflecting barrier at 1, but at 5, we go back to 4, stay at 5 or "wrap around" to 1, each with probability 1/3. Find the new set of stationary probabilities.

2. Consider the Markov model of the shared-memory multiprocessor system in Section 12.1.4. In each part below, your answer will be a function of $q_1, ..., q_m$.

(a) For the case m = 3, find $p_{(2,0,1),(1,1,1)}$.

(b) For the case m = 6, give a compact expression for $p_{(1,1,1,1,1,1),(i,j,k,l,m,n)}$.
Hint: We have an instance of a famous parametric distribution family here.

3. This problem involves the analysis of call centers. This is a subject of much interest in the business world, with there being commercial simulators sold to analyze various scenarios. Here are our assumptions:

- Calls come in according to a Poisson process with intensity parameter λ.
- Call duration is exponentially distributed with parameter η.
- There are always at least b operators in service, and at most b+r.
- Operators work from home, and can be brought into or out of service instantly when needed. They are paid only for the time in service.
- If a call comes in when the current number of operators is larger than b but smaller than b+r, another operator is brought into service to process the call.
- If a call comes in when the current number of operators is b+r, the call is rejected.
- When an operator completes processing a call, and the current number of operators (including this one) is greater than b, then that operator is taken out of service.

Note that this is a birth/death process, with the state being the number of calls currently in the system.

(a) Find approximate closed-form expressions for the π_i for large b+r, in terms of b, r, λ and η. (You should not have any summation symbols.)

(b) Find the proportion of rejected calls, in terms of π_i and b, r, λ and η.

(c) An operator is paid while in service, even if he/she is idle, in which case the wages are "wasted." Express the proportion of wasted time in terms of the π_i and b, r, λ and η.

(d) Suppose b = r = 2, and $\lambda = \eta = 1.0$. When a call completes while we are in state b+1, an operator is sent away. Find the mean time until we make our next summons to the reserve pool.

4. The **bin-packing problem** arises in many computer science applications. Items of various sizes must be placed into fixed-sized bins. The goal is to find a packing arrangement that minimizes unused space. Toward that end, work the following problem.

We are working in one dimension, and have a continuing stream of items arriving, of lengths $L_1, L_2, L_3, ...$ We place the items in the bins in the order of arrival, i.e. without optimizing. We continue to place items in a bin until we encounter an item that will not fit in the remaining space, in which case we go to the next bin.

Suppose the bins are of length 5, and an item has length 1, 2, 3 or 4, with probability 0.25 each. Find the long-run proportion of wasted space.

Hint: Set up a discrete-time Markov chain, with "time" being the number of items packed so far, and the state being the amount of occupied space in the current bin. Define T_n to be 1 or 0, according to whether the n^{th} item causes us to begin packing a new bin, so that the number of bins used by "time" n is $T_1 + ... + T_n$.

5. Suppose we keep rolling a die. Find the mean number of rolls needed to get three consecutive 4s.

Hint: Use the material in Section **??**.

6. A system consists of two machines, with exponentially distributed lifetimes having mean 25.0. There is a single repairperson, but he is not usually on site. When a breakdown occurs, he is summoned (unless he is already on his way or on site), and it takes him a random amount of time to reach the site, exponentially distributed with mean 2.0. Repair time is exponentially distributed with mean 8.0. If after completing a repair the repairperson finds that the other machine needs fixing, he will repair it; otherwise he will leave. Repair is performed on a First Come, First Served schedule. Find the following:

(a) The long-run proportion of the time that the repairperson is on site.

(b) The rate per unit time of calls to the repairperson.

(c) The mean time to repair, i.e. the mean time between a breakdown of a machine and completion of repair of that machine.

(d) The probability that, when two machines are up and one of them goes down, the second machine fails before the repairperson arrives.

7. Consider again the random walk in Section 12.1.1. Find

$$\lim_{n\to\infty} \rho(X_n, X_{n+1}) \tag{12.77}$$

Hint: Apply the Law of Total Expectation to $E(X_n X_{n+1})$.

8. Suppose we model a certain database as follows. New items arrive according to a Poisson process with intensity parameter α. Each item stays in the database for an exponentially distributed amount of time with parameter σ, independently of the other items. Our state at time t is the number of items in the database at that time. Find closed-form expressions for the stationary distribution π and the long-run average size of the database.

9. Consider our machine repair example in Section 11.4, with the following change: The repairperson is offsite, and will not be summoned unless both machines are down. Once the repairperson arrives, she will not leave until both machines are up. So for example, if she arrives and repairs machine B, then while repairing A finds that B has gone down again, she will start work on B immediately after finishing with A. Travel time to the site from the maintenance office is 0. Repair is performed on a First Come, First Served schedule. The time a machine is in working order has an exponential distribution with rate ω, and repair is exponentially distributed with rate ρ. Find the following in terms of ω and ρ:

(a) The long-run proportion of the time that the repairperson is on site.

(b) The rate per unit time of calls to the repairperson.

(c) The mean time to repair, i.e. the mean time between a breakdown of a machine and completion of repair of that machine. (Hint: The best approach is to look at rates. First, find the number of breakdowns per unit time. Then, ask how many of these occur during a time when both machines are up, etc. In each case, what is the mean time to repair for the machine that breaks?)

Chapter 13

Introduction to Queuing Models

Seems like we spend large parts of our lives standing in line (or as they say in New York, standing "on" line). This can be analyzed probabilistically, a subject we will be introduced in this chapter.

13.1 Introduction

Like other areas of applied stochastic processes, queuing theory has a vast literature, covering a huge number of variations on different types of queues. Our tutorial here can only just scratch the surface to this field.

Here is a rough overview of a few of the large categories of queuing theory:

- Single-server queues.
- Networks of queues, including **open** networks (in which jobs arrive from outside the network, visit some of the servers in the network, then leave) and **closed** networks (in which jobs continually circulate within the network, never leaving).
- Non-First Come, First Served (FCFS) service orderings. For example, there are Last Come, First Served (i.e. stacks) and Processor Sharing (which models CPU timesharing).

In this brief introduction, we will not discuss non-FCFS queues, and will only scratch the surface on the other tops.

13.2 M/M/1

The first M here stands for "Markov" or "memoryless," alluding to the fact that arrivals to the queue are Markovian, i.e. interarrivals are i.i.d. exponentially distributed. The second M means that the service times are also i.i.d. exponential. Denote the reciprocal-mean interarrival and service times by λ and μ.

The 1 in M/M/1 refers to the fact that there is a single server. We will assume FCFS job scheduling here, but close inspection of the derivation will show that it applies to some other kinds of scheduling too.

This system is a continuous-time Markov chain, with the state X_t at time t being the number of jobs in the system (not just in the queue but also including the one currently being served, if any).

13.2.1 Steady-State Probabilities

Intuitively the steady-state probabilities π_i will exist only if $\lambda < \mu$. Otherwise jobs would come in faster than they could be served, and the queue would become infinite. So, we assume that $u < 1$, where $u = \frac{\lambda}{\mu}$.

Clearly this is a birth-and-death chain. For state k, the birth rate $\rho_{k,k+1}$ is λ and the death rate $\rho_{k,k-1}$ is μ, k = 0,1,2,... (except that the death rate at state 0 is 0). Using the formula derived for birth/death chains, we have that

$$\pi_i = u^i \pi_0, \ i \geq 0 \tag{13.1}$$

and

$$\pi_0 = \frac{1}{\sum_{j=0}^{\infty} u^j} = 1 - u \tag{13.2}$$

In other words,

$$\pi_i = u^i (1 - u), \ i \geq 0 \tag{13.3}$$

Note by the way that since $\pi_0 = 1 - u$, then u is the *utilization* of the server, i.e. the proportion of the time the server is busy. In fact, this can be seen intuitively: Think of a very long period of time of length t. During this time approximately λt jobs having arrived, keeping the server

busy for approximately $\lambda t \cdot \frac{1}{\mu}$ time. Thus the fraction of time during which the server is busy is approximantely

$$\frac{\lambda t \cdot \frac{1}{\mu}}{t} = \frac{\lambda}{\mu} \tag{13.4}$$

13.2.2 Mean Queue Length

Another way to look at Equation (13.3) is as follows. Let the random variable N have the long-run distribution of X_t, so that

$$P(N = i) = u^i(1 - u),\ i \geq 0 \tag{13.5}$$

Then this says that N+1 has a geometric distribution, with "success" probability 1-u. (N itself is not quite geometrically distributed, since N's values begin at 0 while a geometric distribution begins at 1.)

Thus the long-run average value E(N) for X_t will be the mean of that geometric distribution, minus 1, i.e.

$$EN = \frac{1}{1 - u} - 1 = \frac{u}{1 - u} \tag{13.6}$$

The long-run mean queue length E(Q) will be this value minus the mean number of jobs being served. The latter quantity is $1 - \pi_0 = u$, so

$$EQ = \frac{u^2}{1 - u} \tag{13.7}$$

13.2.3 Distribution of Residence Time/Little's Rule

Let R denote the **residence time** of a job, i.e. the time elapsed from the job's arrival to its exit from the system. Little's Rule says that

$$EN = \lambda ER \tag{13.8}$$

This property holds for a variety of queuing systems, including this one. It can be proved formally, but here is the intuition:

Think of a particular job (in the literature of queuing theory, it is called a "tagged job") at the time it has just exited the system. If this is an "average" job, then it has been in the system for ER amount of time, during which an average of λER new jobs have arrived behind it. These jobs now comprise the total number of jobs in the system, which in the average case is EN.

Applying Little's Rule here, we know EN from Equation (13.6), so we can solve for ER:

$$ER = \frac{1}{\lambda} \frac{u}{1-u} = \frac{1/\mu}{1-u} \tag{13.9}$$

With a little more work, we can find the actual distribution of R, not just its mean. This will enable us to obtain quantities such as Var(R) and P(R > z). Here is our approach:

When a job arrives, say there are N jobs ahead of it, including one in service. Then this job's value of R can be expressed as

$$R = S_{self} + S_{1,resid} + S_2 + ... + S_N \tag{13.10}$$

where S_{self} is the service time for this job, $S_{1,resid}$ is the remaining time for the job now being served (i.e. the residual life), and for $i > 1$ the random variable S_i is the service time for the i^{th} waiting job.

Then the Laplace transform of R, evaluated at say w, is

$$\begin{aligned} E(e^{-wR}) &= E[e^{-w(S_{self}+S_{1,resid}+S_2+...+S_N)}] && (13.11)\\ &= E\left(E[e^{-w(S_{self}+S_{1,resid}+S_2+...+S_N)}|N]\right) && (13.12)\\ &= E[\{E(e^{-wS})\}^{N+1}] && (13.13)\\ &= E[g(w)^{N+1}] && (13.14) \end{aligned}$$

where

$$g(w) = E(e^{-wS}) \tag{13.15}$$

is the Laplace transform of the service variable, i.e. of an exponential distribution with parameter equal to the service rate μ. Here we have made use of these facts:

- The Laplace transform of a sum of independent random variables is the product of their individual Laplace transforms.

- Due to the memoryless property, $S_{1,resid}$ has the same distribution as do the other S_i.
- The distribution of the service times S_i and queue length N observed by our tagged job is the same as the distributions of those quantities at all times, not just at arrival times of tagged jobs. This property can be proven for this kind of queue and many others, and is called PASTA—Poisson Arrivals See Time Averages.

  (Note that the PASTA property is not obvious. On the contrary, given our experience with the Bus Paradox and length-biased sampling in Section 9.4, we should be wary of such things. But the PASTA property does hold and can be proven.)

But that last term in (13.14), $E[g(w)^{N+1}]$, is the generating function of N+1, evaluated at g(w). And we know from Section 13.2.2 that N+1 has a geometric distribution. The generating function for a nonnegative-integer valued random variable K with success probability p is

$$g_K(s) = E(s^K) = \sum_{i=1}^{\infty} s^i (1-p)^{i-1} p = \frac{ps}{1-s(1-p)} \tag{13.16}$$

In (13.14), we have p = 1 -u and s = g(w). So,

$$E(v^{N+1}) = \frac{g(w)(1-u)}{1-u[g(w)]} \tag{13.17}$$

Finally, by definition of Laplace transform,

$$g(w) = E(e^{-wS}) = \int_0^{\infty} e^{-wt} \mu e^{-\mu t} dt = \frac{\mu}{w+\mu} \tag{13.18}$$

So, from (13.11), (13.17) and (13.18), the Laplace transform of R is

$$\frac{\mu(1-u)}{w+\mu(1-u)} \tag{13.19}$$

In principle, Laplace transforms can be inverted, and we could use numerical methods to retrieve the distribution of R from (13.19). But hey, look at that! Equation (13.19) has the same form as (13.18). In other words, we have discovered that R has an exponential distribution too, only with parameter $\mu(1-u)$ instead of μ.

This is quite remarkable. The fact that the service and interarrival times are exponential doesn't mean that everything else will be exponential too, so it is surprising that R does turn out to have an exponential distribution.

It is even more surprising in that R is a sum of independent exponential random variables, as we saw in (13.10), and we know that such sums have Erland distributions. The resolution of this seeming paradox is that the number of terms N in (13.10) is itself random. Conditional on N, R has an Erlang distribution, but unconditionally R has an exponential distribution.

13.3 Multi-Server Models

Here we have c servers, with a common queue. There are many variations.

13.3.1 M/M/c

Here the servers are homogeneous. When a job gets to the head of the queue, it is served by the first available server.

The state is again the number of jobs in the system, including any jobs at the servers. Again it is a birth/death chain, with $u_{i,i+1} = \lambda$ and

$$u_{i,i-1} = \begin{cases} i\mu, & \text{if 0 < i < c} \\ c\mu, & \text{if i} \geq \text{c} \end{cases} \tag{13.20}$$

The solution turns out to be

$$\pi_k = \begin{cases} \pi_0 (\frac{\lambda}{\mu})^k \frac{1}{k!}, & \text{k <}c \\ \pi_0 (\frac{\lambda}{\mu})^k \frac{1}{c! c^{k-c}}, & \text{k} \geq c \end{cases} \tag{13.21}$$

where

$$\pi_0 = \left[\sum_{k=0}^{c-1} \frac{(cu)^k}{k!} + \frac{(cu)^c}{c!} \frac{1}{1-u} \right]^{-1} \tag{13.22}$$

and

$$u = \frac{\lambda}{c\mu} \tag{13.23}$$

Note that the latter quantity is still the utilization per server, using an argument similar to that which led to (13.4).

Recalling that the Taylor series for e^z is $\sum_{k=0}^{\infty} z^k/k!$ we see that

$$\pi_0 \approx e^{-cu} \tag{13.24}$$

13.3.2 M/M/2 with Heterogeneous Servers

Here the servers have different rates. We'll treat the case in which c = 2. Assume $\mu_1 < \mu_2$. When a job reaches the head of the queue, it chooses machine 2 if that machine is idle, and otherwise waits for the first available machine. Once it starts on a machine, it cannot be switched to the other.

Denote the state by (i,j,k), where

- i is the number of jobs at server 1
- j is the number of jobs at server 2
- k is the number of jobs in the queue

The key is to notice that states 111, 112, 113, ... act like the M/M/k queue. This will reduce finding the solution of the balance equations to solving a finite system of linear equations.

For $k \geq 1$ we have

$$(\lambda + \mu_1 + \mu_2)\pi_{11k} = (\mu_1 + \mu_2)\pi_{11,k+1} + \lambda\pi_{11,k-1} \tag{13.25}$$

Collecting terms as in the derivation of the stationary distribution for birth/death processes, (13.25) becomes

$$\lambda(\pi_{11k} - \pi_{11,k-1}) = (\mu_1 + \mu_2)(\pi_{11,k+1} - \pi_{11k}),\ k = 1, 2, ... \tag{13.26}$$

Then we have

$$(\mu_1 + \mu_2)\pi_{11,k+1} - \lambda\pi_{11k} = (\mu_1 + \mu_2)\pi_{11k} - \lambda\pi_{11,k-1} \tag{13.27}$$

So, we now have all the π_{11i}, i = 2,3,... in terms of π_{111} and π_{110}, thus reducing our task to solving a finite set of linear equations, as promised. Here are the rest of the equations:

$$\lambda\pi_{000} = \mu_2\pi_{010} + \mu_1\pi_{100} \tag{13.28}$$

$$(\lambda + \mu_2)\pi_{010} = \lambda\pi_{000} + \mu_1\pi_{110} \tag{13.29}$$

$$(\lambda + \mu_1)\pi_{100} = \mu_2\pi_{110} \tag{13.30}$$

$$(\lambda + \mu_1 + \mu_2)\pi_{110} = \lambda\pi_{010} + \lambda\pi_{100} + (\mu_1 + \mu_2)\pi_{111} \tag{13.31}$$

From (3.91), we have

$$(\mu_1 + \mu_2)\pi_{111} - \lambda\pi_{110} = (\mu_1 + \mu_2)\pi_{110} - \lambda(\pi_{010} + \pi_{100}) \tag{13.32}$$

Look at that last term, $\lambda(\pi_{010} + \pi_{100})$. By adding (13.29) and (13.30), we have that

$$\lambda(\pi_{010} + \pi_{100}) = \lambda\pi_{000} + \mu_1\pi_{110} + \mu_2\pi_{110} - \mu_1\pi_{100} - \mu_2\pi_{010} \tag{13.33}$$

Substituting (13.28) changes (13.33) to

$$\lambda(\pi_{010} + \pi_{100}) = \mu_1\pi_{110} + \mu_2\pi_{110} \tag{13.34}$$

So...(13.32) becomes

$$(\mu_1 + \mu_2)\pi_{111} - \lambda\pi_{110} = 0 \tag{13.35}$$

By induction in (13.27), we have

$$(\mu_1 + \mu_2)\pi_{11,k+1} - \lambda\pi_{11k} = 0,\ k = 1, 2, ... \tag{13.36}$$

and

$$\pi_{11i} = \delta^i\pi_{110},\ i = 0, 1, 2, ... \tag{13.37}$$

where

$$\delta = \frac{\lambda}{\mu_1 + \mu_2} \tag{13.38}$$

$$1 = \sum_{i,j,k} \pi_{ijk} \tag{13.39}$$

$$= \pi_{000} + \pi_{010} + \pi_{100} + \sum_{i=0}^{\infty} \pi_{11i} \tag{13.40}$$

$$= \pi_{000} + \pi_{010} + \pi_{100} + \pi_{110} \sum_{i=0}^{\infty} \delta^i \tag{13.41}$$

$$= \pi_{000} + \pi_{010} + \pi_{100} + \pi_{110} \cdot \frac{1}{1-\delta} \tag{13.42}$$

Finding close-form expressions for the π_i is then straightforward.

13.4 Loss Models

One of the earliest queuing models was M/M/c/c: Markovian interarrival and service times, c servers and a buffer space of c jobs. Any job which arrives when c jobs are already in the system is lost. This was used by telephone companies to find the proportion of lost calls for a bank of c trunk lines.

13.4.1 Cell Communications Model

Let's consider a more modern example of this sort, involving cellular phone systems. (This is an extension of the example treated in K.S. Trivedi, *Probability and Statistics, with Reliability and Computer Science Applications* (second edition), Wiley, 2002, Sec. 8.2.3.2, which is in turn is based on two papers in the *IEEE Transactions on Vehicular Technology*.)

We consider one particular cell in the system. Mobile phone users drift in and out of the cell as they move around the city. A call can either be a **new call**, i.e. a call which someone has just dialed, or a **handoff call**, i.e. a call which had already been in progress in a neighboring cell but now has moved to this cell.

Each call in a cell needs a **channel**.[1] There are n channels available in the cell. We wish to give handoff calls priority over new calls.[2] This is accomplished as follows.

[1] This could be a certain frequency or a certain time slot position.

[2] We would rather give the caller of a new call a polite rejection message, e.g. "No lines available at this time, than suddenly terminate an existing conversation.

The system always reserves g channels for handoff calls. When a request for a new call (i.e. a non-handoff call) arrives, the system looks at X_t, the current number of calls in the cell. If that number is less than n-g, so that there are more than g idle channels available, the new call is accepted; otherwise it is rejected.

We assume that new calls originate from within the cells according to a Poisson process with rate λ_1, while handoff calls drift in from neighboring cells at rate λ_2. Meanwhile, call durations are exponential with rate μ_1, while the time that a call remains within the cell is exponential with rate μ_2.

13.4.1.1 Stationary Distribution

We again have a birth/death process, though a bit more complicated than our earlier ones. Let $\lambda = \lambda_1 + \lambda_2$ and $\mu = \mu_1 + \mu_2$. Then here is a sample balance equation, focused on transitions into (left-hand side in the equation) and out of (right-hand side) state 1:

$$\pi_0 \lambda + \pi_2 2\mu = \pi_1(\lambda + \mu) \tag{13.43}$$

Here's why: How can we enter state 1? Well, we could do so from state 0, where there are no calls; this occurs if we get a new call (rate λ_1) or a handoff call (rate λ_2. In state 2, we enter state 1 if one of the two calls ends (rate μ_1) or one of the two calls leaves the cell (rate μ_2). The same kind of reasoning shows that we leave state 1 at rate $\lambda + \mu$.

As another example, here is the equation for state n-g:

$$\pi_{n-g}[\lambda_2 + (n-g)\mu] = \pi_{n-g+1} \cdot (n-g+1)\mu + \pi_{n-g-1}\lambda \tag{13.44}$$

Note the term λ_2 in (13.44), rather than λ as in (13.43).

Using our birth/death formula for the π_i, we find that

$$\pi_k = \begin{cases} \pi_0 \frac{A^k}{k!}, & \text{k} \leq \text{n-g} \\ \pi_0 \frac{A^{n-g}}{k!} A_1^{k-(n-g)}, & \text{k} \geq \text{n-g} \end{cases} \tag{13.45}$$

where $A = \lambda/\mu$, $A_1 = \lambda_2/\mu$ and

$$\pi_0 = \left[\sum_{k=0}^{n-g-1} \frac{A^k}{k!} + \sum_{k=n-g}^{n} \frac{A^{n-g}}{k!} A_1^{k-(n-g)} \right]^{-1} \tag{13.46}$$

13.4.1.2 Going Beyond Finding the π

One can calculate a number of interesting quantities from the π_i:

- The probability of a handoff call being rejected is π_n.
- The probability of a new call being dropped is

$$\sum_{k=n-g}^{n} \pi_k \tag{13.47}$$

- Since the per-channel utilization in state i is i/n, the overall long-run per-channel utilization is

$$\sum_{i=0}^{n} \pi_i \frac{i}{n} \tag{13.48}$$

- The long-run proportion of accepted calls which are handoff calls is the rate at which handoff calls are accepted, divided by the rate at which calls are accepted:

$$\frac{\lambda_2 \sum_{i=0}^{n-1} \pi_i}{\lambda_1 \sum_{i=0}^{n-g-1} \pi_i + \lambda_2 \sum_{i=0}^{n-1} \pi_i} \tag{13.49}$$

13.5 Nonexponential Service Times

The Markov property is of course crucial to the analyses we made above. Thus dropping the exponential assumption presents a major analytical challenge.

One queuing model which has been found tractable is M/G/1: Exponential interarrival times, general service times, one server. In fact, the mean queue length and related quantities can be obtained fairly easily, as follows.

Consider the residence time R for a tagged job. R is the time that our tagged job must first wait for completion of service of all jobs, if any, which are ahead of it—queued or now in service—plus the tagged job's own service time. Let $T_1, T_2, ...$ be i.i.d. with the distribution of a generic service time random variable S. T_1 represents the service time of the tagged job itself. $T_2, ..., T_N$ represent the service times of the queued jobs, if any.

Let N be the number of jobs in the system, either being served or queued; B be either 1 or 0, depending on whether the system is busy (i.e. N > 0) or not; and $S_{1,resid}$ be the remaining service

time of the job currently being served, if any. Finally, we define, as before, $u = \frac{\lambda}{1/ES}$, the utilization. Note that that implies the EB = u.

Then the distribution of R is that of

$$BS_{1,resid} + (T_1 + ... + T_N) + (1 - B)T_1 \tag{13.50}$$

Note that if N = 0, then $T_1 + ... + T_N$ is considered to be 0, i.e. not present in (13.50).

Then

$$\begin{aligned} E(R) &= uE(S_{1,resid}) + E(T_1 + ... + T_N) + (1 - u)ET_1 && (13.51)\\ &= uE(S_{1,resid}) + E(N)E(S) + (1 - u)ES && (13.52)\\ &= uE(S_{1,resid}) + \lambda E(R)E(S) + (1 - u)ES && (13.53) \end{aligned}$$

The last equality is due to Little's Rule. Note also that we have made use of the PASTA property here, so that the distribution of N is the same at arrival times as general times.

Then

$$E(R) = \frac{uE(S_{1,resid})}{1 - u} + ES \tag{13.54}$$

Note that the two terms here represent the mean residence time as the mean queuing time plus the mean service time.

So we must find $E(S_{1,resid})$. This is just the mean of the remaining-life distribution which we saw in Section 15.6 of our unit on renewal theory. Then

$$\begin{aligned} E(S_{1,resid}) &= \int_0^\infty t \frac{1 - F_S(t)}{ES}\, dt && (13.55)\\ &= \frac{1}{ES} \int_0^\infty t \int_t^\infty f_S(u)\, du\, dt && (13.56)\\ &= \frac{1}{ES} \int_0^\infty f_S(u) \int_0^u t\, dt\, du && (13.57)\\ &= \frac{1}{2ES} E(S^2) && (13.58) \end{aligned}$$

So,

$$E(R) = \frac{uE(S^2)}{2ES(1-u)} + ES \tag{13.59}$$

What is remarkable about this famous formula is that E(R) depends not only on the mean service time but also on the variance. This result, which is not so intuitively obvious at first glance, shows the power of modeling. We might observe the dependency of E(R) on the variance of service time empirically if we do simulation, but here is a compact formula that shows it for us.

13.6 Reversed Markov Chains

We can get insight into some kinds of queuing systems by making use of the concepts of **reversed** Markov chains, which involve "playing the Markov chain backward," just as we could play a movie backward.

Consider a continuous-time, irreducible, positive recurrent Markov chain X(t).[3] For any fixed time τ (typically thought of as large), define the **reversed** version of X(t) as Y(t) = X(τ-t), for $0 \leq t \leq \tau$. We will discuss a number of properties of reversed chains. These properties will enable what mathematicians call "soft analysis" of some Markov chains, especially those related to queues. This term refers to short, simple, elegant proofs or derivations.

13.6.1 Markov Property

The first property to note is that Y(t) is a Markov chain! Here is our first chance for soft analysis.

The "hard analysis" approach would be to start with the definition, which in continuous time would be that

$$P\left(Y(t) = k | Y(u), u \leq s\right) = P\left(Y(t) = k | Y(s)\right) \tag{13.60}$$

for all $0 < s < t$ and all k, using the fact that X(t) has the same property. That would involve making substitutions in Equation (13.60) like Y(t) = X(τ-t), etc.

[3]Recall that a Markov chain is irreducible if it is possible to get from each state to each other state in a finite number of steps, and that the term *positive recurrent* means that the chain has a long-run state distribution π. Also, concerning our assumption here of continuous time, we should note that there are discrete-time analogs of the various points we'll make below.

But it is much easier to simply observe that the Markov property holds if and only if, conditional on the present, the past and the future are independent. Since that property holds for X(t), it also holds for Y(t) (with the roles of the "past" and the "future" interchanged).

13.6.2 Long-Run State Proportions

Clearly, if the long-run proportion of the time X(t) = k is π_i, the same long-run proportion will hold for Y(t). This of course only makes sense if you think of larger and larger τ.

13.6.3 Form of the Transition Rates of the Reversed Chain

Let $\tilde{\rho}_{ij}$ denote the number of transitions from state i to state j per unit time in the reversed chain. That number must be equal to the number of transitions from j to i in the original chain. Therefore,

$$\pi_i \tilde{\rho}_{ij} = \pi_j \rho_{ji} \tag{13.61}$$

This gives us a formula for the $\tilde{\rho}_{ij}$:

$$\tilde{\rho}_{ij} = \frac{\pi_j}{\pi_i} \rho_{ji} \tag{13.62}$$

13.6.4 Reversible Markov Chains

In some cases, the reversed chain has the same probabilistic structure as the original one! Note carefully what that would mean. In the continuous-time case, it would mean that $\tilde{\rho}_{ij} = \rho_{ij}$ for all i and j, where the $\tilde{\rho}_{ij}$ are the transition rates of Y(t).[4] If this is the case, we say that X(t) is **reversible**.

That is a very strong property. An example of a chain which is not reversible is the tree-search model in Section 12.3.5.[5] There the state space consists of all the nonnegative integers, and transitions were possible from states n to n+1 and from n to 0. Clearly this chain is <u>not</u> reversible, since we can go from n to 0 in one step but not vice versa.

[4] Note that for a continuous-time Markov chain, the transition rates do indeed uniquely determined the probabilistic structure of the chain, not just the long-run state proportions. The short-run behavior of the chain is also determined by the transition rates, and at least in theory can be calculated by solving differential equations whose coefficients make use of those rates.

[5] That is a discrete-time example, but the principle here is the same.

13.6.4.1 Conditions for Checking Reversibility

Equation (13.61) shows that the original chain X(t) is reversible if and only if

$$\pi_i \rho_{ij} = \pi_j \rho_{ji} \tag{13.63}$$

for all i and j. These equations are called the **detailed balance equations**, as opposed to the general **balance equations**,

$$\sum_{j \neq i} \pi_j \rho_{ji} = \pi_i \lambda_i \tag{13.64}$$

which are used to find the π values. Recall that (13.64) arises from equating the flow into state i with the flow out of it. By contrast, Equation (13.63) equates the flow into i from a particular state j to the flow from i to j. Again, that is a much stronger condition, so we can see that most chains are not reversible. However, a number of important ones are reversible, as we'll see.

For example, consider birth/death chains. Here, the only cases in which ρ_{rs} is nonzero are those in which $|i - j| = 1$. Now, Equation (11.19) in our derivation of π for birth/death chains is exactly (13.63)! So we see that birth/death chains are reversible.

More generally, equations (13.63) may not be so easy to check, since for complex chains we may not be able to find closed-form expressions for the π values. Thus it is desirable to have another test available for reversibility. One such test is **Kolmogorov's Criterion**:

> The chain is reversible if and only if for any **loop** of states, the product of the transition rates is the same in both the forward and backward directions.

For example, consider the loop $i \to j \to k \to i$. Then we would check whether $\rho_{ij}\rho_{jk}\rho_{ki} = \rho_{ik}\rho_{kj}\rho_{ji}$.

Technically, we do have to check *all* loops. However, in many cases it should be clear that just a few loops are representative, as the other loops have the same structure.

Again consider birth/death chains. Kolmogorov's Criterion trivially shows that they are reversible, since any loop involves a path which is the same path when traversed in reverse.

13.6.4.2 Making New Reversible Chains from Old Ones

Since reversible chains are so useful (when we are lucky enough to have them), a very useful trick is to be able to form new reversible chains from old ones. The following two properties are very handy in that regard:

(a) Suppose U(t) and V(t) are reversible Markov chains, and define W(t) to be the tuple [U(t),V(t)]. Then W(t) is reversible.

(b) Suppose X(t) is a reversible Markov chain, and A is an irreducible subset of the state space of the chain, with long-run state distribution π. Define a chain W(t) with transition rates $\rho\prime_{ij}$ for $i \in A$, where $\rho\prime_{ij} = \rho_{ij}$ if $j \in A$ and $\rho\prime_{ij} = 0$ otherwise. Then W(t) is reversible, with long-run state distribution given by

$$\pi\prime_i = \frac{\pi_i}{\sum_{j \in A} \pi_j} \tag{13.65}$$

13.6.4.3 Example: Distribution of Residual Life

In Section 15.6.2, we used Markov chain methods to derive the age distribution at a fixed observation point in a renewal process. From remarks made there, we know that residual life has the same distribution. This could be proved similarly, at some effort, but it comes almost immediately from reversibility considerations. After all, the residual life in the reversed process is the age in the original process.

13.6.4.4 Example: Queues with a Common Waiting Area

Consider two M/M/1 queues, with chains G(t) and H(t), with independent arrival streams but having a common waiting area, with jobs arriving to a full waiting area simply being lost.[6]

First consider the case of an infinite waiting area. Let u_1 and u_2 be the utilizations of the two queues, as in (13.3). G(t) and H(t), being birth/death processes, are reversible. Then by property (a) above, the chain [G(t),H(t)] is also reversible. Long-run proportion of the time that there are m jobs in the first queue and n jobs in the second is

$$\pi_{mn} = (1 - u_1)^m u_1 (1 - u_2)^n u_2 \tag{13.66}$$

for m,n = 0,1,2,3,...

Now consider what would happen if these two queues were to have a common, finite waiting area. Denote the amount of space in the waiting area by w. The new process is the restriction of the original process to a subset of states A as in (b) above. (The set A will be precisely defined below.) It is easily verified from the Kolmogorov Criterion that the new process is also reversible.

[6] Adapted from Ronald Wolff, *Stochastic Modeling and the Theory of Queues* Prentice Hall, 1989.

Recall that the state m in the original queue U(t) is the number of jobs, including the one in service if any. That means the number of jobs waiting is $(m-1)^+$, where $x^+ = \max(x, 0)$. That means that for our new system, with the common waiting area, we should take our subset A to be

$$\{(m,n) : m,n \geq 0, (m-1)^+ + (n-1)^+ \leq w\} \tag{13.67}$$

So, by property (b) above, we know that the long-run state distribution for the queue with the finite common waiting area is

$$\pi_{mn} = \frac{1}{a}(1-u_1)^m u_1 (1-u_2)^n u_2 \tag{13.68}$$

where

$$a = \sum_{(i,j) \in A} (1-u_1)^i u_1 (1-u_2)^j u_2 \tag{13.69}$$

In this example, reversibility was quite useful. It would have been essentially impossible to derive (13.68) algebraically. And even if intuition had suggested that solution as a guess, it would have been quite messy to verify the guess.

13.6.4.5 Closed-Form Expression for π for Any Reversible Markov Chain

(Adapted from Ronald Nelson, *Probability, Stochastic Processes and Queuing Theory*, Springer-Verlag, 1995.)

Recall that most Markov chains, especially those with infinite state spaces, do not have closed-form expressions for the steady-state probabilities. But we can always get such expressions for reversible chains, as follows.

Choose a fixed state s, and find paths from s to all other states. Denote the path to i by

$$s = j_{i1} \to j_{i2} \to ... \to j_{im_i} = i \tag{13.70}$$

Define

$$\psi_i = \begin{cases} 1, & \text{i = s} \\ \Pi_{k=1}^{m_i} r_{ik}, & \text{i} \neq \text{s} \end{cases} \tag{13.71}$$

where

$$r_{ik} = \frac{\rho(j_{ik}, j_{i,k+1})}{\rho(j_{i,k+1}, j_{i,k})} \tag{13.72}$$

Then the steady-state probabilities are

$$\pi_i = \frac{\psi_i}{\sum_k \psi_k} \tag{13.73}$$

You may notice that this looks similar to the derivation for birth/death processes, which as has been pointed out, are reversible.

13.7 Networks of Queues

13.7.1 Tandem Queues

Let's first consider an M/M/1 queue. As mentioned earlier, this is a birth/death process, thus reversible. This has an interesting and very useful application, as follows.

Think of the times at which jobs *depart* this system, i.e. the times at which jobs finish service. In the reversed process, these times are *arrivals*. Due to the reversibility, that means that the distribution of departure times is the same as that of arrival times. In other words:

- Departures from this system behave as a Poisson process with rate λ.

Also, let the initial state X(0) be distributed according to the steady-state probabilities π.[7] Due to the PASTA property of Poisson arrivals, the distribution of the system state at arrival times is the same as the distribution of the system state at nonrandom times t. Then by reversibility, we have that:

- The state distribution at departure times is the same as at nonrandom times.

And finally, noting as in Section 13.6.1 that, given X(t), the states $\{X(s), s \leq t\}$ of the queue before time t are statistically independent of the arrival process after time t, reversibility gives us that:

[7] Recall Section 12.1.2.4.

- Given t, the departure process before time t is statistically independent of the states $\{X(s), s \geq t\}$ of the queue after time t.

Let's apply that to **tandem** queues, which are queues acting in series. Suppose we have two such queues, with the first, $X_1(t)$ feeding its output to the second one, $X_2(t)$, as input. Suppose the input into $X_1(t)$ is a Poisson process with rate λ, and service times at both queues are exponentially distributed, with rates μ_1 and μ_2.

$X_1(t)$ is an M/M/1 queue, so its steady-state probabilities for $X_1(t)$ are given by Equation (13.3), with $u = \lambda/\mu_1$.

By the first bulleted item above, we know that the input into $X_2(t)$ is also Poisson. Therefore, $X_2(t)$ also is an M/M/1 queue, with steady-state probabilities as in Equation (13.3), with $u = \lambda/\mu_2$.

Now, what about the joint distribution of $[X_1(t), X_2(t)]$? The third bulleted item above says that the input to $X_2(t)$ up to time t is independent of $\{X_1(s), s \geq t\}$. So, using the fact that we are assuming that $X_1(0)$ has the steady-state distribution, we have that

$$P[X_1(t) = i, X_2(t) = j] = (1 - u_1)u_1^i P[X_2(t) = j] \tag{13.74}$$

Now letting $t \to \infty$, we get that the long-run probability of the vector $[X_1(t), X_2(t)]$ being equal to (i,j) is

$$(1 - u_1)u_1^j(1 - u_2)u_2^j \tag{13.75}$$

In other words, the steady-state distribution for the vector has the two components of the vector being independent.

Equation (13.75) is called a **product form solution** to the balance equations for steady-state probabilities.

By the way, the vector $[X_1(t), X_2(t)]$ is *not* reversible.

13.7.2 Jackson Networks

The tandem queues discussed in the last section comprise a special case of what are known as **Jackson networks**. Once again, there exists an enormous literature of Jackson and other kinds of queuing networks. The material can become very complicated (even the notation is very complex), and we will only present an introduction here. Our presentation is adapted from I. Mitrani, *Modelling of Computer and Communcation Systems*, Cambridge University Press, 1987.

Our network consists of N nodes, and jobs move from node to node. There is a queue at each node, and service time at node i is exponentially distributed with mean $1/\mu_i$.

13.7.2.1 Open Networks

Each job originally arrives externally to the network, with the arrival rate at node i being γ_i. After moving among various nodes, the job will eventually leave the network. Specifically, after a job completes service at node i, it moves to node j with probability q_{ij}, where

$$\sum_j q_{ij} < 1 \tag{13.76}$$

reflecting the fact that the job will leave the network altogether with probability $1 - \sum_j q_{ij}$.[8] It is assumed that the movement from node to node is memoryless.

As an example, you may wish to think of movement of packets among routers in a computer network, with the packets being jobs and the routers being nodes.

Let λ_i denote the total traffic rate into node i. By the usual equating of flow in and flow out, we have

$$\lambda_i = \gamma_i + \sum_{j=1}^{N} \lambda_j q_{ji} \tag{13.77}$$

Note the in Equations (13.77), the knowns are γ_i and the q_{ji}. We can solve this system of linear equations for the unknowns, λ_i.

The utilization at node i is then $u_i = \lambda_i/\mu_i$, as before. Jackson's Theorem then says that in the long run, node i acts as an M/M/1 queue with that utilization, and that the nodes are independent in the long run:[9]

$$\lim_{t\to\infty} P[X_1(t) = i_1, ..., X_N(t) = i_N] = \Pi_{i=1}^{N}(1 - u_i)u^i \tag{13.78}$$

So, again we have a product form solution.

[8] By the way, q_{ii} can be nonzero, allowing for feedback loops at nodes.

[9] We do not present the proof here, but it really is just a matter of showing that the distribution here satisfies the balance equations.

Let L_i denote the average number of jobs at node i. From Equation (13.6), we have $L_i = u_i/(1-u_i)$. Thus the mean number of jobs in the system is

$$L = \sum_{i=1}^{N} \frac{u_i}{1-u_i} \tag{13.79}$$

From this we can get the mean time that jobs stay in the network, W: From Little's Rule, $L = \gamma W$, so

$$W = \frac{1}{\gamma} \sum_{i=1}^{N} \frac{u_i}{1-u_i} \tag{13.80}$$

where $\gamma = \gamma_1 + ... + \gamma_N$ is the total external arrival rate.

Jackson networks are not generally reversible. The reversed versions of Jackson networks are worth studying for other reasons, but we cannot pursue them here.

13.7.3 Closed Networks

In a closed Jackson network, we have for all i, $\gamma_i = 0$ and

$$\sum_j q_{ij} = 1 \tag{13.81}$$

In other words, jobs never enter or leave the network. There have been many models like this in the computer performance modeling literature. For instance, a model might consist of some nodes representing CPUs, some representing disk drives, and some representing users at terminals.

It turns out that we again get a product form solution.[10] The notation is more involved, so we will not present it here.

Exercises

1. Investigate the robustness of the M/M/1 queue model with respect to the assumption of exponential service times, as follows. Suppose the service time is actually uniformly distributed on (0,c), so that the mean service time would be c/2. Assume that arrivals do follow the exponential model, with mean interarrival time 1.0. Find the mean residence time, using (13.9), and compare

[10] This is confusing, since the different nodes are now not independent, due to the fact that the number of jobs in the overall system is constant.

it to the true value obtained from (13.59). Do this for various values of c, and graph the two curves using R.

2. Many mathematical analyses of queuing systems use **finite source** models. There are always a fixed number j of jobs in the system. A job queues up for the server, gets served in time S, then waits a random time W before queuing up for the server again.

A typical example would be a file server with j clients. The time W would be the time a client does work before it needs to access the file server again.

(a) Use Little's Rule, on two or more appropriately chosen boxes, to derive the following relation:

$$ER = \frac{jES}{U} - EW \tag{13.82}$$

where R is residence time (time spent in the queue plus service time) in one cycle for a job and U is the utilization fraction of the server.

(b) Set up a continuous time Markov chain, assuming exponential distributions for S and W, with state being the number of jobs currently at the server. Derive closed-form expressions for the π_i.

3. Consider the following variant of an M/M/1 queue: Each customer has a certain amount of patience, varying from one customer to another, exponentially distributed with rate η. When a customer's patience wears out while the customer is in the queue, he/she leaves (but not if his/her job is now in service). Arrival and service rates are λ and ν, respectively.

(a) Express the π_i in terms of λ, ν and η.

(b) Express the proportion of lost jobs as a function of the π_i, λ, ν and η.

4. A shop has two machines, with service time in machine i being exponentially distributed with rate μ_i, i = 1,2. Here $\mu_1 > \mu_2$. When a job reaches the head of the queue, it chooses machine 1 if that machine is idle, and otherwise waits for the first available machine. If when a job finishes on machine 1 there is a job in progress at machine 2, the latter job will be transferred to machine 1, getting priority over any queued jobs. Arrivals follow the usual Poisson process, parameter λ.

(a) Find the mean residence time.

(b) Find the proportion of jobs that are originally assigned to machine 2.

Chapter 14

Describing "Failure"

Here we will study the distributions of times until event occurrence. There are numerous applications, such as:

- Medical research: E.g. time of recurrence of a disease.
- Software reliability: E.g. time until the next bug is discovered.
- Software development processes: Time until a new team member is added to an open source project.
- Hardware reliability: Time until failure.
- Marketing: E.g. time until your customer moves to some other provider.
- Employment discrimination litigation: E.g. time until a worker is fired.

Here in this chapter, we study the mathematical underpinnings.

14.1 Hazard Functions

In addition to density functions, another useful description of a continuous distribution is its **hazard function**.[1] Again think of the lifetimes of light bulbs, not necessarily assuming an exponential distribution. Intuitively, the hazard function states the likelihood of a bulb failing in the next short interval of time, given that it has lasted up to now. To understand this, let's review a certain property of the exponential distribution family.

[1] This can be done for discrete random variables too.

14.1.1 Basic Concepts

Suppose the lifetimes of light bulbs L were discrete. Suppose a particular bulb has already lasted 80 hours. The probability of it failing in the next hour could be written in terms of the pmf and cdf of the distribution of lifetimes of bulbs:

$$P(L = 81|L > 80) = \frac{P(L = 81 \text{ and } L > 80)}{P(L > 80)} = \frac{P(L = 81)}{P(L > 80)} = \frac{p_L(81)}{1 - F_L(80)} \tag{14.1}$$

In general, for discrete L, we define its **hazard function** as

$$h_L(i) = \frac{p_L(i)}{1 - F_L(i - 1)} \tag{14.2}$$

By analogy, for continuous L we define

$$h_L(t) = \frac{f_L(t)}{1 - F_L(t)} \tag{14.3}$$

Again, the interpretation is that $h_L(t)$ is the likelihood of the item failing very soon after t, given that it has lasted t amount of time. The probability of failing within the next δ amount of time is approximately $h_L(t)\delta$, for small δ.

Note carefully that the word "failure" here should not be taken literally. In our Davis railroad crossing example in Chapter 9, "failure" means that the train ends—a "failure" which those of us who are waiting will welcome!

Since we know that exponentially distributed random variables are memoryless, we would expect intuitively that their hazard functions are constant. We can verify this by evaluating (14.3) for an exponential density with parameter λ; sure enough, the hazard function is constant, with value λ.

The reader should verify that in contrast to an exponential distribution's constant failure rate, a uniform distribution has an increasing failure rate (IFR). Some distributions have decreasing failure rates, while most have non-monotone rates.

Hazard function models have been used extensively in software testing. Here "failure" is the discovery of a bug, and with quantities of interest include the mean time until the next bug is discovered, and the total number of bugs.

Some parametric families of distributions have strictly increasing failure rates (IFR). Some have strictly decreasing failure rates (DFR). People have what is called a "bathtub-shaped" hazard function. It is high near 0 (reflecting infant mortality) and after, say, 70, but is low and rather flat in between.

Chapter 18

Transform Methods

We often use the idea of **transform** functions. For example, you may have seen **Laplace transforms** in a math or engineering course. The functions we will see here differ from this by just a change of variable.

Though in the form used here they involve only univariate distributions, their applications are often multivariate, as will be the case here.

18.1 Generating Functions

Let's start with the **generating function**. For any nonnegative-integer valued random variable V, its generating function is defined by

$$g_V(s) = E(s^V) = \sum_{i=0}^{\infty} s^i p_V(i),\ 0 \leq s \leq 1 \tag{18.1}$$

For instance, suppose N has a geometric distribution with parameter p, so that $p_N(i) = (1-p)p^{i-1}$, i = 1,2,... Then

$$g_N(s) = \sum_{i=1}^{\infty} s^i \cdot (1-p)p^{i-1} = \frac{1-p}{p} \sum_{i=1}^{\infty} s^i \cdot p^i = \frac{1-p}{p} \frac{ps}{1-ps} = \frac{(1-p)s}{1-ps} \tag{18.2}$$

Why restrict s to the interval [0,1]? The answer is that for $s > 1$ the series in (18.1) may not converge. for $0 \leq s \leq 1$, the series does converge. To see this, note that if s = 1, we just get the

sum of all probabilities, which is 1.0. If a nonnegative s is less than 1, then s^i will also be less than 1, so we still have convergence.

One use of the generating function is, as its name implies, to generate the probabilities of values for the random variable in question. In other words, if you have the generating function but not the probabilities, you can obtain the probabilities from the function. Here's why: For clarify, write (18.1) as

$$g_V(s) = P(V = 0) + sP(V = 1) + s^2 P(V = 2) + ... \tag{18.3}$$

From this we see that

$$g_V(0) = P(V = 0) \tag{18.4}$$

So, we can obtain P(V = 0) from the generating function. Now differentiating (18.1) with respect to s, we have

$$\begin{aligned} g'_V(s) &= \frac{d}{ds}\left[P(V = 0) + sP(V = 1) + s^2 P(V = 2) + ...\right] \\ &= P(V = 1) + 2sP(V = 2) + ... \end{aligned} \tag{18.5}$$

So, we can obtain P(V = 2) from $g'_V(0)$, and in a similar manner can calculate the other probabilities from the higher derivatives.

18.2 Moment Generating Functions

The generating function is handy, but it is limited to discrete random variables. More generally, we can use the **moment generating function**, defined for any random variable X as

$$m_X(t) = E[e^{tX}] \tag{18.6}$$

for any t for which the expected value exists.

That last restriction is anathema to mathematicians, so they use the characteristic function,

$$\phi_X(t) = E[e^{itX}] \tag{18.7}$$

which exists for any t. However, it makes use of pesky complex numbers, so we'll stay clear of it here.

Differentiating (18.6) with respect to t, we have

$$m'_X(t) = E[Xe^{tX}] \tag{18.8}$$

We see then that

$$m'_X(0) = EX \tag{18.9}$$

So, if we just know the moment-generating function of X, we can obtain EX from it. Also,

$$m''_X(t) = E(X^2 e^{tX}) \tag{18.10}$$

so

$$m''_X(0) = E(X^2) \tag{18.11}$$

In this manner, we can for various k obtain $E(X^k)$, the **k***th* **moment** of X, hence the name.

18.3 Transforms of Sums of Independent Random Variables

Suppose X and Y are independent and their moment generating functions are defined. Let Z = X+Y. then

$$m_Z(t) = E[e^{t(X+Y)}] = E[e^{tX}e^{tY}] = E(e^{tX}) \cdot E(e^{tY}) = m_X(t)m_Y(t) \tag{18.12}$$

In other words, the mgf of the sum is the product of the mgfs! This is true for other transforms, by the same reasoning.

Similarly, it's clear that the mgf of a sum of three independent variables is again the product of their mgfs, and so on.

18.4 Example: Network Packets

As an example, suppose say the number of packets N received on a network link in a given time period has a Poisson distribution with mean μ, i.e.

$$P(N = k) = \frac{e^{-\mu}\mu^k}{k!}, k = 0, 1, 2, 3, ... \tag{18.13}$$

18.4.1 Poisson Generating Function

Let's first find its generating function.

$$g_N(t) = \sum_{k=0}^{\infty} t^k \frac{e^{-\mu}\mu^k}{k!} = e^{-\mu} \sum_{k=0}^{\infty} \frac{(\mu t)^k}{k!} = e^{-\mu+\mu t} \tag{18.14}$$

where we made use of the Taylor series from calculus,

$$e^u = \sum_{k=0}^{\infty} u^k/k! \tag{18.15}$$

18.4.2 Sums of Independent Poisson Random Variables Are Poisson Distributed

Supposed packets come in to a network node from two independent links, with counts N_1 and N_2, Poisson distributed with means μ_1 and μ_2. Let's find the distribution of $N = N_1 + N_2$, using a transform approach.

From Section 18.3:

$$g_N(t) = g_{N_1}(t) g_{N_2}(t) = e^{-\nu+\nu t} \tag{18.16}$$

where $\nu = \mu_1 + \mu_2$.

But the last expression in (18.16) is the generating function for a Poisson distribution too! And since there is a one-to-one correspondence between distributions and transforms, we can conclude that N has a Poisson distribution with parameter ν. We of course knew that N would have mean ν but did not know that N would have a Poisson distribution.

So: A sum of two independent Poisson variables itself has a Poisson distribution. By induction, this is also true for sums of k independent Poisson variables.

18.5 Other Uses of Transforms

Transform techniques are used heavily in queuing analysis, including for models of computer networks. The techniques are also used extensively in modeling of hardware and software reliability.

Transforms also play key roles in much of theoretical probability, the Central Limit Theorems[1] being a good example. Here's an outline of the proof of the basic CLT, assuming the notation of Section 8.19:

First rewrite Z as

$$Z = \sum_{i=1}^{n} \frac{X_i - m}{v\sqrt{n}} \tag{18.17}$$

Then work with the characteristic function of Z:

$$\begin{aligned} c_Z(t) &= E(e^{itZ}) \quad \text{(def.)} && (18.18)\\ &= \Pi_{i=1}^{n} E[e^{it(X_i-m)/(v\sqrt{n})}] \quad \text{(indep.)} && (18.19)\\ &= \Pi_{i=1}^{n} E[e^{it(X_1-m)/(v\sqrt{n})}] \quad \text{(ident. distr.)} && (18.20)\\ &= [g(\frac{it}{\sqrt{n}})]^n && (18.21) \end{aligned}$$

where g(s) is the characteristic function of $(X_1 - m)/v$, i.e.

$$g(s) = E[e^{is \cdot \frac{X_1-m}{v}}] \tag{18.22}$$

Now expand (18.21) in a Taylor series around 0, and use the fact that $g'(0)$ is the expected value of $(X_1 - m)/v$, which is 0:

$$\begin{aligned} [g(\frac{t}{\sqrt{n}})]^n &= \left[1 - \frac{t^2}{2n} + o(\frac{t^2}{n})\right]^n && (18.23)\\ &\to e^{-t^2/2} \text{ as } n \to \infty && (18.24) \end{aligned}$$

where we've also used the famous fact that $(1 - s/n)^n$ converges to e^{-s} as $n \to \infty$.

[1] The plural is used here because there are many different versions, which for instance relax the condition that the summands be independent and identically distributed.

But (18.24) is the density of N(0,1), so we have proved the Central Limit Theorem.

Exercises

1. Use transform methods to derive some properties of the Poisson family:

(a) Show that for any Poisson random variable, its mean and variance are equal.

(b) Suppose X and Y are independent random variables, each having a Poisson distribution. Show that $Z = X + Y$ again has a Poisson distribution.

2. In our ordinary coins which we use every day, each one has a slightly different probability of heads, which we'll call H. Say H has the distribution $N(0.5, 0.03^2)$. We choose a coin from a batch at random, then toss it 10 times. Let N be the number of heads we get. Find $Var(N)$.

3. Suppose the number N of bugs in a certain number of lines of code has a Poisson distribution, with parameter L, where L varies from one programmer to another. Show that Var(N) = EL + Var(L).

4. Let X denote the number we obtain when we roll a single die once. Let $G_X(s)$ denote the generating function of X.

(a) Find $G_X(s)$.

(b) Suppose we roll the die 5 times, and let T denote the total number of dots we get from the 5 rolls. Find $G_T(s)$.

Chapter 22

General Statistical Estimation and Inference

Earlier, we often referred to certain estimators as being "natural." For example, if we are estimating a population mean, an obvious choice of estimator would be the sample mean. But in many applications, it is less clear what a "natural" estimate for a population quantity of interest would be. We will present general methods for estimation in this section.

We will also discuss advanced methods of inference.

22.1 General Methods of Parametric Estimation

Let's begin with a simple motivating example.

22.1.1 Example: Guessing the Number of Raffle Tickets Sold

You've just bought a raffle ticket, and find that you have ticket number 68. You check with a couple of friends, and find that their numbers are 46 and 79. Let c be the total number of tickets. How should we estimate c, using our data 68, 46 and 79?

Let $X_1, X_2, ..., X_n$ denote the ticket numbers we are aware of. In the above case, $n = 3$, with $X_1 = 68$ etc. We will treat the X_i as a random sample from 1,2,...,c. This assumption should be carefully considered. Recall that this means that $X_1, X_2, ..., X_n$ are i.i.d. with a (discrete) uniform distribution on 1,2....,c.

If we know, say, that the three friends bought their tickets right after they became available.

This might mean the X_i are nearer to 1 than to c, in which case the assumption of the uniform distribution may not be very good. Or, suppose we know that the three friends were standing close to each other in line. Then the independence assumption might be dubious.

And even if not, and even if the X_i are considered a sample with the uniformity assumption satisfied, it would technically be what is called a *simple* random sample (Section 19.1.1), since the "sampling" is done without replacement; no two people get the same ticket.

So, in practice one would need to consider how good our assumption is that we have a random sample. If $n << c$, the various probabilities are virtually identical for sampling with and without replacement, so that may not be an issue, so we would just worry about the uniformity.

But let's suppose that has been resolved, and then look at how we can estimate c.

22.1.2 Method of Moments

One approach, a very intuitive one, is the Method of Momentsi (MM). It is less commonly used than the other method we'll cover, Maximum Likelihood, but is much easier to explain. Also, in recent years, MM has become more popular in certain fields, such as random graph models (Section 2.13).

22.1.2.1 Example: Lottery Model

Let X be distributed the same as the X_i, i.e. it is uniformly distributed on 1,2,...,c.

Note first that

$$E(X) = \frac{c+1}{2} \tag{22.1}$$

Let's solve for c:

$$c = 2EX - 1 \tag{22.2}$$

We know that we can use

$$\overline{X} = \frac{1}{n} \sum_{i=1}^{n} X_i \tag{22.3}$$

to estimate EX, so by (22.2), $2\overline{X} - 1$ is an intuitive estimate of c. Thus we take our estimator for c to be

$$\widehat{c} = 2\overline{X} - 1 \tag{22.4}$$

This estimator is called the Method of Moments estimator of c.

22.1.2.2 General Method

Say our sample is $X_1, ..., X_n; a, and$ we have k parameters to estimate, $\theta_1, ..., \theta_k$. Our goal is to come up with estimators $\widehat{\theta_1}, ..., \widehat{\theta_k}$.

Here is the procedure:

- For each $i = 1, ..., k$, write $E(X^k)$ as a function of the θ_j,

$$E(X^i) = g_i(\theta_1, ..., \theta_k) \tag{22.5}$$

- On the left side of (22.5), replace $E(X^i)$ by the corresponding sample moment

$$\frac{1}{n}\sum_{r=1}^{n} X_r^i \tag{22.6}$$

- On the right side of (22.5), replace each θ_s by $\widehat{\theta_s}$.
- Solve for the $\widehat{\theta_s}$

In the raffle example, we had k = 1, $\theta_1 = c$, $g_1(c) = (c+1)/2$ and so on.

22.1.3 Method of Maximum Likelihood

Another method, much more commonly used, is called the **Method of Maximum Likelihood**.

22.1.3.1 Example: Raffle Model

In our example above, it means asking the question, "What value of c would have made our data—68, 46, 79—most likely to happen?" Well, let's find what is called the **likelihood**, i.e. the

probability of our particular data values occurring:

$$L = P(X_1 = 68, X_2 = 46, X_3 = 79) = \begin{cases} (\frac{1}{c})^3, & \text{if } c \geq 79 \\ 0, & \text{otherwise} \end{cases} \tag{22.7}$$

Now keep in mind that c is a fixed, though unknown constant. It is not a random variable. What we are doing here is just asking "What if" questions, e.g. "If c were 85, how likely would our data be? What about c = 91?"

Well then, what value of c maximizes (22.7)? Clearly, it is c = 79. Any smaller value of c gives us a likelihood of 0. And for c larger than 79, the larger c is, the smaller (22.7) is. So, our maximum likelihood estimator (MLE) is 79. In general, if our sample size in this problem were n, our MLE for c would be

$$\check{c} = \max_i X_i \tag{22.8}$$

Note that in our earlier discussion of whether our data can be treated as a random sample, we raised the issue of the independence assumption. Note that this assumption in fact *was* used for $\check{c}$, while it was not for $\widehat{c}$.

22.1.3.2 General Procedure

Say again our random sample is $X_1, ..., X_n$ and we have k parameters to estimate, $\theta_1, ..., \theta_k$. Our goal is to come up with estimators $\widehat{\theta_1}, ..., \widehat{\theta_k}$.

Denote the pmf or density of our X_i as $f(t; \theta_1, ..., \theta_k)$. Then our estimators $\widehat{\theta_i}$ are the values that maximize the likelihood

$$\Pi_{i=1}^n f(X_i; \theta_1, ..., \theta_k) \tag{22.9}$$

with respect to the θ_j.

For "smooth" problems, i.e. ones in which the likelihood is a differentiable function of the parameters, aaximization involves taking the derivatives with respect to the θ_j, setting them to 0, then solving for the θ_j; the solutions are the $\widehat{\theta_j}$.

Since it is easier to take derivatives of sums than products, typically one maximizes the log likelihood

$$\sum_{i=1}^n \ln[f(X_i; \theta_1, ..., \theta_k)] \tag{22.10}$$

22.1.4 Example: Estimation of the Parameters of a Gamma Distribution

As another example, suppose we have a random sample $X_1, ..., X_n$ from a gamma distribution.

$$f_X(t) = \frac{1}{\Gamma(c)} \lambda^c t^{c-1} e^{-\lambda t},\ t > 0 \tag{22.11}$$

for some unknown c and λ. How do we estimate c and λ from the X_i?

22.1.4.1 Method of Moments

Let's try the Method of Moments, as follows. We have two population parameters to estimate, c and λ, so we need to involve two moments of X. That could be EX and $E(X^2)$, but here it would more conveniently be EX and Var(X). We know from our previous unit on continuous random variables, Chapter 7, that

$$EX = \frac{c}{\lambda} \tag{22.12}$$

$$Var(X) = \frac{c}{\lambda^2} \tag{22.13}$$

In our earlier notation, this would be r = 2, $\theta_1 = c$, $\theta_2 = \lambda$ and $g_1(c, \lambda) = c/\lambda$ and $g_2(c, \lambda) = c/\lambda^2$.

Switching to sample analogs and estimates, we have

$$\frac{\widehat{c}}{\widehat{\lambda}} = \overline{X} \tag{22.14}$$

$$\frac{\widehat{c}}{\widehat{\lambda}^2} = s^2 \tag{22.15}$$

Dividing the two quantities yields

$$\widehat{\lambda} = \frac{\overline{X}}{s^2} \tag{22.16}$$

which then gives

$$\widehat{c} = \frac{\overline{X}^2}{s^2} \tag{22.17}$$

22.1.4.2 MLEs

What about the MLEs of c and λ? Remember, the X_i are continuous random variables, so the likelihood function, i.e. the analog of (22.7), is the product of the density values:

$$L = \Pi_{i=1}^{n} \left[\frac{1}{\Gamma(c)} \lambda^c {X_i}^{c-1} e^{-\lambda X_i} \right] \tag{22.18}$$

$$= [\lambda^c/\Gamma(c)]^n \left(\Pi_{i=1}^{n} X_i\right)^{c-1} e^{-\lambda \sum_{i=1}^{n} X_i} \tag{22.19}$$

In general, it is usually easier to maximize the log likelihood (and maximizing this is the same as maximizing the original likelihood):

$$l = (c-1) \sum_{i=1}^{n} \ln(X_i) - \lambda \sum_{i=1}^{n} X_i + nc \ln(\lambda) - n \ln(\Gamma(c)) \tag{22.20}$$

One then takes the partial derivatives of (22.20) with respect to c and λ, and sets the derivatives to zero. The solution values, $\check{c}$ and $\check{\lambda}$, are then the MLEs of c and λ. Unfortunately, in this case, these equations do not have closed-form solutions. So the equations must be solved numerically. (In fact, numerical methods are needed even more in this case, because finding the derivative of $\Gamma(c)$ is not easy.)

22.1.5 R's mle() Function

R provides a function, **mle()**, for finding MLEs in mathematically intractable situations such as the one in the last section.

Note: The function is in the **stats4** library, so run

```
> library(stats4)
```

first.

Here's an example in that context. We'll simulate some data from a gamma distribution with given parameter values, then pretend we don't know those, and find the MLEs from the data:

```
> n <- 1000
> x <- rgamma(n,shape=2,rate=1)  # Erlang, r = 2, lambda is 1

# function to compute negative log likelihood
ll <- function(c,lambda) {
```

```
   loglik <- (c-1) * sum(log(x)) - sum(x)*lambda + n*c*log(lambda) -
      n*log(gamma(c))
   -loglik
}

summary(mle(minuslogl=ll,start=list(c=1.5,lambda=2)))
Maximum likelihood estimation

Call:
mle(minuslogl = ll, start = list(c = 1, lambda = 1))

Coefficients:
       Estimate Std. Error
c      2.147605 0.08958823
lambda 1.095344 0.05144393

-2 log L: 3072.181
```

How did this work? The main task we have is to write a function that calculates the negative log likelihood, with that function's arguments will be the parameters to be estimated. We defined our function and named it **ll()**.[1]

Fortunately for us, **mle()** calculates the derivatives numerically too, so we didn't need to specify them in the log likelihood function. (Needless to say, this function thus cannot be used in a problem in which derivatives cannot be used, such as the lottery example above.)

We also need to supply **mle()** with initial guesses for the parameters. That's done in the **start** argument. I more or less arbitrarily chose these values. You may have to experiment, though, as some sets of initial values may not result in convergence.

The standard errors of the estimated parameters are also printed out, enabling the formation of confidence intervals and significance tests. (See Section 20.5 for the case of confidence intervals.) In fact, you can get the estimated covariance matrix for the vector of estimated parameters. In our case here:

```
> mleout <- mle(minuslogl=ll,start=list(c=1.5,lambda=2))
> solve(mleout@details$hessian)
                 c      lambda
c      0.008026052 0.004093526
lambda 0.004093526 0.002646478
```

By the way, there were also some warning messages, due to the fact that during the iterative maximization process, some iterations generated guesses for λ were 0 or near it, causing problems with **log()**.

Maximizing the likelihood meant minimizing the negative log likelihood. Why did the authors of R

[1] Note that in R, **log()** calculates the natural logarithm by default.

write it this way, and in fact report -2 times the log likelihood? The answer is that this is related to a significance testing procedure, the *likelihood ratio test*, a matter beyond the scope of this book.

Note that if you are using **mle()** with a parametric family for which R supplies a density function, it's quite convenient to simply use that in our likelihood function code:

```
> ll <- function(c,lambda) -sum(dgamma(x,shape=c,rate=lambda,log=TRUE))
> summary(mle(minuslogl=ll,start=list(c=1.5,lambda=2)))
Maximum likelihood estimation

Call:
mle(minuslogl = ll, start = list(c = 1.5, lambda = 2))

Coefficients:
       Estimate Std. Error
c      2.147605 0.08958823
lambda 1.095344 0.05144393

-2 log L: 3072.181
```

R made things so convenient that we didn't even need to take the logs ourselves.

22.1.6 R's gmm() Function

A function to do Method of Moments estimation is also available, in R's **gmm** package on CRAN. The package actually does more, something called Generalized Method of Moments, but we'll see how to use it in elementary form here.

The work is done by the eponymous function **gmm()**. In our usage here, the call form is

```
gmm(data,momentftn{,start)}
```

where **data** is our data in matrix or vector form, **momentftn** specifies the moments, and **start** is our initial guess for the iterative solution process.

The function **momentftn()** specifies how to compute the differences between the estimated values of the population moments (in terms of the parameter estimates) and the corresponding sample powers. During the computation, **gmm()** will set the means of these differences to 0, iterating until convergence.

22.1.6.1 Example: Bodyfat Data

Our data set will be **bodyfat**, which is included in the **mfp** package, with measurements on 252 men. The first column of this data frame, **brozek**, is the percentage of body fat, which when converted to a proportion is in (0,1). That makes it a candidate for fitting a beta distribution (Section 7.6.5).

Here is our **momentftn()**:

```
g <- function(th,x) {
   t1 <- th[1]
   t2 <- th[2]
   t12 <- t1 + t2
   meanb <- t1 / t12
   m1 <- meanb - x
   m2 <- t1*t2 / (t12^2 * (t12+1)) - (x - meanb)^2
   f <- cbind(m1,m2)
   return(f)
}
```

Here the argument **th** ("theta") will be the MM estimates (at any given iteration) of the population parameters, in this case of α and β.

Now look at the line

```
m1 <- meanb - x
```

The value **meanb** is our estimate in the currently iteration of the mean of the beta distribution, while **x** is our data. The **gmm()** function, in calling our **g()**, will calculate the average of **m1** over all our data, and then in setting that average to 0, we are equating our parametric estimate of the population mean to our sample mean — exactly what MM is supposed to do.

Let's try it:

```
> library(mfp)
> data(bodyfat)
> gmm(g,x,c(alpha=0.1,beta=0.1))
Method
 twoStep

Objective function value: 2.559645e-10

 alpha  beta
 4.6714 19.9969
```

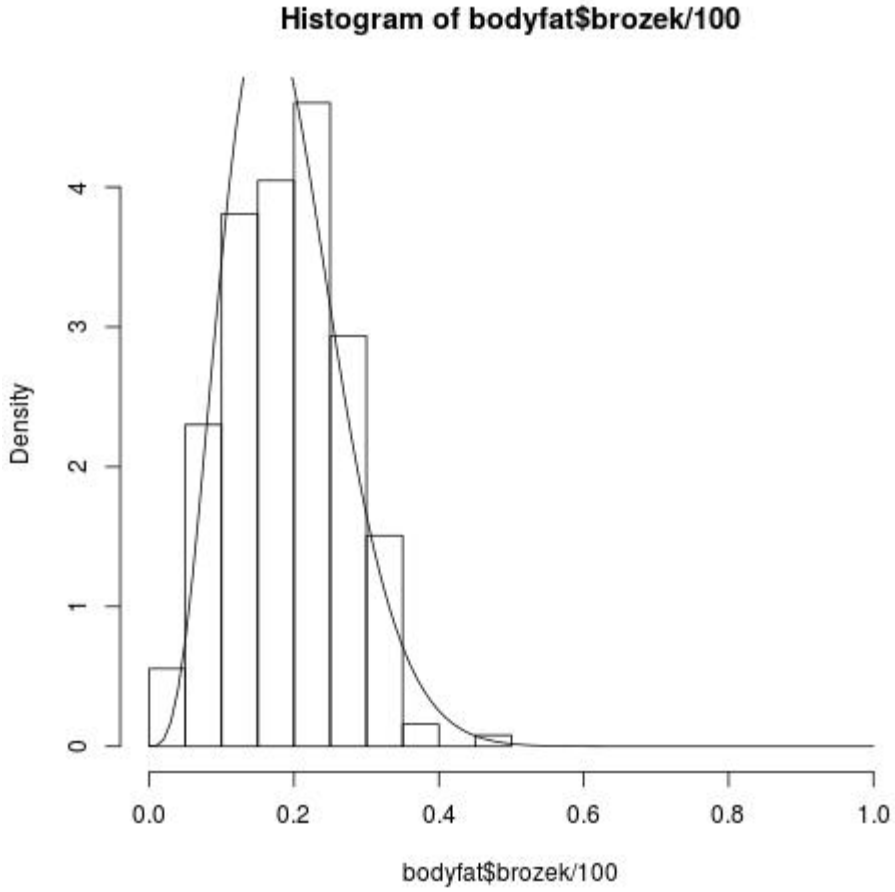

Figure 22.1: Fitting a Beta Density

```
Convergence code = 0
> hist(bodyfat$brozek/100,xlim=c(0,1),
    probability=TRUE)
> curve(dbeta(x,4.67,20.00),add=TRUE)
```

The result is seen in Figure 22.1. At least visually, the fit seems pretty good.

22.1.7 More Examples

Suppose $f_W(t) = ct^{c-1}$ for t in (0,1), with the density being 0 elsewhere, for some unknown $c > 0$. We have a random sample $W_1, ..., W_n$ from this density.

Let's find the Method of Moments estimator.

$$EW = \int_0^1 tct^{c-1}\, dt = \frac{c}{c+1} \tag{22.21}$$

So, set

$$\overline{W} = \frac{\widehat{c}}{\widehat{c}+1} \tag{22.22}$$

yielding

$$\widehat{c} = \frac{\overline{W}}{1-\overline{W}} \tag{22.23}$$

What about the MLE?

$$L = \Pi_{i=1}^n cW_i^{c-1} \tag{22.24}$$

so

$$l = n \ln c + (c-1) \sum_{i=1}^n \ln W_i \tag{22.25}$$

Then set

$$0 = \frac{n}{\widehat{c}} + \sum_{i=1}^n \ln W_i \tag{22.26}$$

and thus

$$\widehat{c} = -\frac{1}{\frac{1}{n}\sum_{i=1}^n \ln W_i} \tag{22.27}$$

As in Section 22.1.3, not every MLE can be determined by taking derivatives. Consider a continuous analog of the example in that section, with $f_W(t) = \frac{1}{c}$ on (0,c), 0 elsewhere, for some $c > 0$.

The likelihood is

$$\left(\frac{1}{c}\right)^n \tag{22.28}$$

as long as

$$c \geq \max_i W_i \tag{22.29}$$

and is 0 otherwise. So,

$$\widehat{c} = \max_i W_i \tag{22.30}$$

as before.

Now consider a different problem. Suppose the random variable X is equal to 1, 2 and 3, with probabilities c, c and 1-2c. The value c is thus a population parameter. We have a random sample $X_1, ..., X_n$ from this population. Let's find the Method of Moments Estimator of c, and its bias.

First,

$$EX = c \cdot 1 + c \cdot 2 + (1 - 2c) \cdot 3 = 3 - 3c \tag{22.31}$$

Thus

$$c = (3 - EX)/3 \tag{22.32}$$

and so set

$$\hat{c} = (3 - \overline{X})/3 \tag{22.33}$$

Next,

$$
\begin{aligned}
E\hat{c} &= E[(3-\overline{X})/3) && (22.34)\\
&= \frac{1}{3}\cdot(3-E\overline{X}) && (22.35)\\
&= \frac{1}{3}[3-EX] && (22.36)\\
&= \frac{1}{3}[3-(3-3c)] && (22.37)\\
&= c && (22.38)
\end{aligned}
$$

On average, not too high and not too low; we say the *bias* is 0.

22.1.8 Asymptotic Properties

Most of the inference methods in this book use large-sample, i.e. *asymptotic* techniques, based on the Central Limit Theorem. Let's look at the asymptotic properties of Method of Moments and Maximum Likelihood estimators.

22.1.8.1 Consistency

We say that an estimator $\widehat{\nu}$ of some parameter ν is **consistent** if, with probability 1,

$$\lim_{n\to\infty}\widehat{\nu}=\nu \tag{22.39}$$

where n is the sample size. In other words, as the sample size grows, the estimator eventually converges to the true population value. Clearly, in most situations consistency is a minimal property we want our estimators to have.

Of course here $\overline{X}$ is a consistent estimator of EX, and (22.6) is a consistent estimator of $E(X^i)$. So, as long as the $g_i()$ in (22.6) are continous etc., we see that the Method of Moments gives us consistent estimators.

Showing consistency for MLEs is a little more involved. To start, recall (22.10). Its derivative with respect to θ_j is

$$\sum_{i=1}^{n}\frac{f'(X_i;\theta_1,...,\theta_k)}{f(X_i;\theta_1,...,\theta_k)} \tag{22.40}$$

For each value of $\theta = (\theta_1, ..., \theta_k)'$, this will converge to

$$E\left(\frac{f'(X;\theta)}{f(X;\theta)}\right) \tag{22.41}$$

Let η denote the true population value of θ, with $\widehat{\theta}$ being the MLE. The latter is the root of (22.40),

$$0 = \sum_{i=1}^{n} \frac{f'(X_i;\widehat{\theta})}{f(X_i;\widehat{\theta})} \tag{22.42}$$

Moreover, in for instance the continuous case (we'll be implicitly assuming some mathematical conditions here, e.g. that roots are uniqe),

$$\begin{aligned} E\left(\frac{f'(X;\eta)}{f(X;\eta)}\right) &= \int_{-\infty}^{\infty} f'(t;\eta)\ dt && (22.43)\\ &= \frac{d}{dt}\int_{-\infty}^{\infty} f(t;\eta)\ dt && (22.44)\\ &= \frac{d}{dt}1 && (22.45)\\ &= 0 && (22.46) \end{aligned}$$

Due to the uniqeness of the roots, we see that $\widehat{theta}$, the root of (22.40) must converge to η, the root of (22.41).

22.1.8.2 Approximate Confidence Intervals

Usually we are not satisfied with simply forming estimates (called **point estimates**). We also want some indication of how accurate these estimates are, in the form of confidence intervals (**interval estimates**).

In many special cases, finding confidence intervals can be done easily on an *ad hoc* basis. Look, for instance, at the Method of Moments Estimator in Section 22.1.2. Our estimator (22.4) is a linear function of $\overline{X}$, so we easily obtain a confidence interval for c from one for EX.

Another example is (22.27). Taking the limit as $n \to \infty$ the equation shows us (and we could verify) that

$$c = \frac{1}{E[\ln W]} \tag{22.47}$$

Defining $X_i = \ln W_i$ and $\overline{X} = (X_1 + ... + X_n)/$, we can obtain a confidence interval for EX in the usual way. We then see from (22.47) that we can form a confidence interval for c by simply taking the reciprocal of each endpoint of the interval, and swapping the left and right endpoints.

What about in general? For the Method of Moments case, our estimators are functions of the sample moments, and since the latter are formed from sums and thus are asymptotically normal, the delta method (Section 32.2) can be used to show that our estimators are asymptotically normal and to obtain asymptotic variances for them.

There is a well-developed asymptotic theory for MLEs, which under certain conditions shows asymptotic normality with a certain asymptotic variance, thus enabling confidence intervals. The theory also establishes that MLEs are in a certain sense optimal among all estimators. We will not pursue this here, but will note that **mle()** does give standard errors for the estimates, thus enabling the formation of confidence intervals.

22.2 Bias and Variance

The notions of **bias** and **variance** play central roles in the evaluation of goodness of estimators.

22.2.1 Bias

This bowl of porridge is not too big, not too small, but just right—from the children's story, *Goldilocks* (paraphrased)

Definition 42 *Suppose $\hat{\theta}$ is an estimator of θ. Then the* **bias** *of $\hat{\theta}$ is*

$$bias = E(\hat{\theta}) - \theta \tag{22.48}$$

If the bias is 0, we say that the estimator is **unbiased**.

So, if $\hat{\theta}$ is an unbiased estimator of θ, then its average value over all possible samples is not too high, not too low, but just right.

At first that would seem to be a "must have" property for any estimator. But it's very important to note that, in spite of the pejorative-sounding name, bias is not an inherently bad property for an estimator to have. Indeed, most good estimators are at least slightly biased.[2] We'll explore this

[2]Typically, though, the amount of bias will go to 0 as the sample size goes to infinity. That is the case for most consistent estimators (Sec. 22.1.2, though technically it is not implied by consistency; if a sequence of random variables converges to a limit, their expected values do not necessarily converge to that limit, or converge at all).

in the next section.

22.2.2 Why Divide by n-1 in s^2?

It should be noted that it is customary in (19.23) to divide by n-1 instead of n, for reasons that are largely historical. Here's the issue:

If we divide by n, as we have been doing, then it turns out that s^2 is biased.

$$E(s^2) = \frac{n-1}{n} \cdot \sigma^2 \tag{22.49}$$

Think about this in the Davis people example, once again in the notebook context. Remember, here n is 1000, and each line of the notebook represents our taking a different random sample of 1000 people. Within each line, there will be entries for W_1 through W_{1000}, the weights of our 1000 people, and for $\overline{W}$ and s. For convenience, let's suppose we record that last column as s^2 instead of s.

Now, say we want to estimate the population variance σ^2. As discussed earlier, the natural estimator for it would be the sample variance, s^2. What (22.49) says is that after looking at an infinite number of lines in the notebook, the average value of s^2 would be just...a...little...bit...too...small. All the s^2 values would average out to $0.999\sigma^2$, rather than to σ^2. We might say that s^2 has a little bit more tendency to underestimate σ^2 than to overestimate it.

So, (22.49) implies that s^2 is a biased estimator of the population variance σ^2, with the amount of bias being

$$\frac{n-1}{n} \cdot \sigma^2 - \sigma^2 = -\frac{1}{n} \cdot \sigma^2 \tag{22.50}$$

Let's prove (22.49). As before, let $W_1, ..., W_n$ be a random sample from some population. So, $EW_i = \mu$ and $Var(W_i) = \sigma^2$, where again μ and σ^2 are the population mean and variance.

It will be more convenient to work with ns^2 than s^2, since it will avoid a lot of dividing by n. So, write

$$
\begin{aligned}
ns^2 &= \sum_{i=1}^{n}(W_i - \overline{W})^2 \quad \text{(def.)} && (22.51)\\
&= \sum_{i=1}^{n}\left[(W_i - \mu) + (\mu - \overline{W})\right]^2 \quad \text{(alg.)} && (22.52)\\
&= \sum_{i=1}^{n}(W_i - \mu)^2 + 2(\mu - \overline{W})\sum_{i=1}^{n}(W_i - \mu) + n(\mu - \overline{W})^2 \quad \text{(alg.)} && (22.53)
\end{aligned}
$$

But that middle sum is

$$\sum_{i=1}^{n}(W_i - \mu) = \sum_{i=1}^{n} W_i - n\mu = n\overline{W} - n\mu \tag{22.54}$$

So,

$$ns^2 = \sum_{i=1}^{n}(W_i - \mu)^2 - n(\overline{W} - \mu)^2 \tag{22.55}$$

Now let's take the expected value of (22.55). First,

$$
\begin{aligned}
E\left(\sum_{i=1}^{n}(W_i - \mu)^2\right) &= \sum_{i=1}^{n} E[(W_i - \mu)^2] \quad \text{(E is lin.)} && (22.56)\\
&= \sum_{i=1}^{n} E[(W_i - EW_i)^2] \quad (W_i \text{ distr. as pop.}) && (22.57)\\
&= \sum_{i=1}^{n} Var(W_i) \quad \text{(def. of Var())} && (22.58)\\
&= \sum_{i=1}^{n} \sigma^2 \quad (W_i \text{ distr. as pop.}) && (22.59)\\
&= n\sigma^2 && (22.60)
\end{aligned}
$$

Also,

$$
\begin{aligned}
E[(\overline{W}-\mu)^2] &= E[(\overline{W}-E\overline{W})^2] \quad ((19.8)) && (22.61)\\
&= Var(\overline{W}) \quad \text{(def. of Var())} && (22.62)\\
&= \frac{\sigma^2}{n} \quad (19.13) && (22.63)
\end{aligned}
$$

Applying these last two findings to (22.55), we get (22.49).

$$E(s^2) = \frac{n-1}{n}\sigma^2 \tag{22.64}$$

The earlier developers of statistics were bothered by this bias, so they introduced a "fudge factor" by dividing by n-1 instead of n in (19.23). We will call that $\tilde{s}^2$:

$$\tilde{s}^2 = \frac{1}{n-1}\sum_{i=1}^{n}(W_i-\overline{W})^2 \tag{22.65}$$

This is the "classical" definition of sample variance, in which we divide by n-1 instead of n.

The R functions **var()** and **sd()** calculate the versions of s^2 and s, respectively, that have a divisor of n-1. In other words, **var()** calculates (22.65), and **sd()** computes its square root.

22.2.2.1 But in This Book, We Divide by n, not n-1 Anyway

But we will use n. After all, when n is large—which is what we are assuming by using the Central Limit Theorem in most of the inference machinery here—it doesn't make any appreciable difference. Clearly it is not important in our Davis example, or our bus simulation example.

Moreover, speaking generally now rather than necessarily for the case of s^2 there is no particular reason to insist that an estimator be unbiased anyway. An alternative estimator may have a little bias but much smaller variance, and thus might be preferable.

And anyway, even though the classical version of s^2, i.e. $\tilde{s}^2$, is an unbiased estimator for σ^2, $\tilde{s}$ is not an unbiased estimator for σ, the population standard deviation (see below). Since we typically have use for $\tilde{s}$ rather than for $\tilde{s}^2$—in (20.4), for example—you can see that unbiasedness is not such an important property after all.

Let's show that $\tilde{s}$ is biased. Recalling the shortcut formula $Var(U) = E(U^2) - (EU)^2$, we have

$$\begin{aligned} 0 &< Var(\tilde{s}) && (22.66)\\ &= E[\tilde{s}^2] - [E\tilde{s}]^2 && (22.67)\\ &= \sigma^2 - [E\tilde{s}]^2 && (22.68) \end{aligned}$$

since $\tilde{s}^2$ is an unbiased estimator of σ^2. So,

$$E\tilde{s} < \sigma \tag{22.69}$$

and $\tilde{s}$ is biased downward.[3]

So, $\tilde{s}$, the standard estimator of σ, is indeed biased, as are many other standard estimators of various quantities. It would be futile to insist on unbiasedness as a criterion of the goodness of an estimator.

22.2.3 Example of Bias Calculation: Max from U(0,c)

Let's find the bias of the estimator (22.30).

The bias is $E\widehat{c} - c$. To get $E\widehat{c}$ we need the density of that estimator, which we get as follows:

$$\begin{aligned} P(\widehat{c} \leq t) &= P(\text{all } W_i \leq t) \ \ \text{(definition)} && (22.70)\\ &= \left(\frac{t}{c}\right)^n \ \ \text{(density of } W_i\text{)} && (22.71) \end{aligned}$$

So,

$$f_{\widehat{c}}(t) = \frac{n}{c^n} t^{n-1} \tag{22.72}$$

Integrating against t, we find that

$$E\widehat{c} = \frac{n}{n+1}\ c \tag{22.73}$$

[3]The reader may wonder why we have strict inequality in (22.66). But although it tis rue that Var(U) can be 0, you'll recall that that occurs only when U is constant. Here it would mean that $\tilde{s}$ is constant. This in turn would mean that all the W_i in (22.65) are identical, with probability 1.0—which would mean the population random variable W is constant, e.g. everyone in Davis has the same weight. So, other that in that absurd situation, the inequality in (22.66) will indeed be strict.

So the bias is c/(n+1), not bad at all.

22.2.4 Example of Bias Calculation: Gamma Family

Let us find via simulation, the bias of the Method of Moments Estimator of the parameter λ for the family of gamma distributions. (The estimator was derived in Section 22.1.4.)

```
lambbias <- function(r,lamb,n,nreps) {
   lambhat <- vector(length=nreps)
   unfudge <- (n-1) / n
   for (i in 1:nreps) {
      x <- rgamma(n,shape=r,rate=lamb)
      xbar <- mean(x)
      s2 <- var(x) * unfudge
      lambhat[i] <- xbar / s2
   }
   mean(lambhat) - lamb
}
```

22.2.5 Tradeoff Between Variance and Bias

Consider a general estimator Q of some population value b. Then a common measure of the quality (of course there are many others) of the estimator Q is the **mean squared error** (MSE),

$$E[(Q-b)^2] \tag{22.74}$$

Of course, the smaller the MSE, the better.

One can break (22.74) down into variance and (squared) bias components, as follows:[4]

$$\begin{aligned}
MSE(Q) &= E[(Q-b)^2] \text{ (definition)} && (22.75)\\
&= E[\{(Q-EQ)+(EQ-b)\}^2] \text{ (algebra)} && (22.76)\\
&= E[(Q-EQ)^2] + 2E\left[(Q-EQ)(EQ-b)\right] + E[(EQ-b)^2] \text{ (E props.)} && (22.77)\\
&= E[(Q-EQ)^2] + E[(EQ-b)^2] \text{ (factor out constant EQ-b)} && (22.78)\\
&= Var(Q) + (EQ-b)^2 \text{ (def. of Var(), fact that EQ-b is const.)} && (22.79)\\
&= \text{variance + squared bias} && (22.80)
\end{aligned}$$

[4] In reading the following derivation, keep in mind that EQ and b are constants.

In other words, in discussing the accuracy of an estimator—especially in comparing two or more candidates to use for our estimator—the average squared error has two main components, one for variance and one for bias. In building a model, these two components are often at odds with each other; we may be able to find an estimator with smaller bias but more variance, or vice versa.

We also see from (22.80) that a little bias in an estimator may be quite tolerable, as long as the variance is low. This is good, because as mentioned earlier, most estimators are in fact biased.

These point will become central in Chapters 26 and 27.

22.3 Simultaneous Inference Methods

Events of small probability happen all the time, because there are so many of them—Jim Sutton, old Cal Poly economics professor

Suppose in our study of heights, weights and so on of people in Davis, we are interested in estimating a number of different quantities, with our forming a confidence interval for each one. Though our confidence level for each one of them will be 95%, our *overall* confidence level will be less than that. In other words, we cannot say we are 95% confident that all the intervals contain their respective population values.

In some cases we may wish to construct confidence intervals in such a way that we can say we are 95% confident that all the intervals are correct. This branch of statistics is known as **simultaneous inference** or **multiple inference**.

(The same issues apply to significance testing, but we will focus on confidence intervals here.)

In this age of Big Data, simultaneous inference is a major issue. We may have hundreds of variables, so the chances of getting spurious results are quite high.

Usually this kind of methodology is used in the comparison of several **treatments**. This term originated in the life sciences, e.g. comparing the effectiveness of several different medications for controlling hypertension, it can be applied in any context. For instance, we might be interested in comparing how well programmers do in several different programming languages, say Python, Ruby and Perl. We'd form three groups of programmers, one for each language, with say 20 programmers per group. Then we would have them write code for a given application. Our measurement could be the length of time T that it takes for them to develop the program to the point at which it runs correctly on a suite of test cases.

Let T_{ij} be the value of T for the j^{th} programmer in the i^{th} group, i = 1,2,3, j = 1,2,...,20. We would then wish to compare the three "treatments," i.e. programming languages, by estimating $\mu_i = ET_{i1}$, i = 1,2,3. Our estimators would be $U_i = \sum_{j=1}^{20} T_{ij}/20$, i = 1,2,3. Since we are comparing the three population means, we may not be satisfied with simply forming ordinary 95% confidence

intervals for each mean. We may wish to form confidence intervals which *jointly* have confidence level 95%.[5]

Note very, very carefully what this means. As usual, think of our notebook idea. Each line of the notebook would contain the 60 observations; different lines would involve different sets of 60 people. So, there would be 60 columns for the raw data, three columns for the U_i. We would also have six more columns for the confidence intervals (lower and upper bounds) for the μ_i. Finally, imagine three more columns, one for each confidence interval, with the entry for each being either Right or Wrong. A confidence interval is labeled Right if it really does contain its target population value, and otherwise is labeled Wrong.

Now, if we construct individual 95% confidence intervals, that means that in a given Right/Wrong column, in the long run 95% of the entries will say Right. But for simultaneous intervals, we hope that within a line we see three Rights, and 95% of all lines will have that property.

In our context here, if we set up our three intervals to have individual confidence levels of 95%, their simultaneous level will be $0.95^3 = 0.86$, since the three confidence intervals are independent. Conversely, if we want a simultaneous level of 0.95, we could take each one at a 98.3% level, since $0.95^{\frac{1}{3}} \approx 0.983$.

However, in general the intervals we wish to form will not be independent, so the above "cube root method" would not work. Here we will give a short introduction to more general procedures.

Note that "nothing in life is free." If we want simultaneous confidence intervals, they will be wider.

Another reason to form simultaneous confidence intervals is that it gives you "license to browse," i.e. to rummage through the data looking for interesting nuggets.

22.3.1 The Bonferonni Method

One simple approach is **Bonferonni's Inequality**:

Lemma 43 *Suppose $A_1, ..., A_g$ are events. Then*

$$P(A_1 \text{ or } ... \text{ or} A_g) \leq \sum_{i=1}^{g} P(A_i) \tag{22.81}$$

[5]The word *may* is important here. It really is a matter of philosophy as to whether one uses simultaneous inference procedures.

You can easily see this for g = 2:

$$P(A_1 \text{ or } A_2) = P(A_1) + P(A_2) - P(A_1 \text{ and } A_2) \leq P(A_1) + P(A_2) \tag{22.82}$$

One can then prove the general case by mathematical induction.

Now to apply this to forming simultaneous confidence intervals, take A_i to be the event that the i^{th} confidence interval is incorrect, i.e. fails to include the population quantity being estimated. Then (23.1) says that if, say, we form two confidence intervals, each having individual confidence level (100-5/2)%, i.e. 97.5%, then the overall collective confidence level for those two intervals is at least 95%. Here's why: Let A_1 be the event that the first interval is wrong, and A_2 is the corresponding event for the second interval. Then

$$\begin{aligned}
\text{overall conf. level} &= P(\text{not } A_1 \text{ and not } A_2) && (22.83)\\
&= 1 - P(A_1 \text{ or } A_2) && (22.84)\\
&\geq 1 - P(A_1) - P(A_2) && (22.85)\\
&= 1 - 0.025 - 0.025 && (22.86)\\
&= 0.95 && (22.87)
\end{aligned}$$

22.3.2 Scheffe's Method

The Bonferonni method is unsuitable for more than a few intervals; each one would have to have such a high individual confidence level that the intervals would be very wide. Many alternatives exist, a famous one being **Scheffe's method**.[6] The large-sample version we'll use here is:

Theorem 44 *Suppose $R = (R_1, ..., R_k)'$ has an approximately multivariate normal distribution, with mean vector $\mu = (\mu_i)$ and covariance matrix $\Sigma = (\sigma_{ij})$. Let $\widehat{\Sigma}$ be a* **consistent** *estimator of Σ, meaning that it converges in probability to Σ as the sample size goes to infinity.*

For any constants $c_1, ..., c_k$, consider linear combinations of the R_i,

$$\sum_{i=1}^{k} c_i R_i \tag{22.88}$$

[6]The name is pronounced "sheh-FAY."

which estimate

$$\sum_{i=1}^{k} c_i \mu_i \tag{22.89}$$

Form the confidence intervals

$$\sum_{i=1}^{k} c_i R_i \pm \sqrt{\chi^2_{\alpha;k}} s(c_1, ..., c_k) \tag{22.90}$$

where

$$[s(c_1, ..., c_k)]^2 = (c_1, ..., c_k)^T \widehat{\Sigma} (c_1, ..., c_k) \tag{22.91}$$

and where $\chi^2_{\alpha;k}$ is the upper-α percentile of a chi-square distribution with k degrees of freedom.[7]

Then all of these intervals (for infinitely many values of the c_i!) have simultaneous confidence level $1 - \alpha$.

By the way, if we are interested in only constructing confidence intervals for **contrasts**, i.e. c_i having the property that $\Sigma_i c_i = 0$, we the number of degrees of freedom reduces to k-1, thus producing narrower intervals.

Just as in Section 22.2.2 we avoided the t-distribution, here we have avoided the F distribution, which is used instead of ch-square in the "exact" form of Scheffe's method.

22.3.3 Example

For example, again consider the Davis heights example in Section 20.6. Suppose we want to find approximate 95% confidence intervals for two population quantities, μ_1 and μ_2. These correspond to values of c_1, c_2 of (1,0) and (0,1). Since the two samples are independent, $\sigma_{12} = 0$. The chi-square value is 5.99,[8] so the square root in (23.10) is 3.46. So, we would compute (20.4) for $\overline{X}$ and then for $\overline{Y}$, but would use 3.46 instead of 1.96.

This actually is not as good as Bonferonni in this case. For Bonferonni, we would find two 97.5% confidence intervals, which would use 2.24 instead of 1.96.

[7] Recall that the distribution of the sum of squares of g independent N(0,1) random variables is called **chi-square with g degrees of freedom**. It is tabulated in the R statistical package's function **qchisq()**.

[8] Obtained from R via **qchisq(0.95,2)**.

Scheffe's method is too conservative if we just are forming a small number of intervals, but it is great if we form a lot of them. Moreover, it is very general, usable whenever we have a set of approximately normal estimators.

22.3.4 Other Methods for Simultaneous Inference

There are many other methods for simultaneous inference. It should be noted, though, that many of them are limited in scope, and quite a few of them are oriented only toward significance testing, rather than confidence intervals. In any event, they are beyond the scope of this book.

22.4 Bayesian Methods

Everyone is entitled to his own opinion, but not his own facts—Daniel Patrick Moynihan, senator from New York, 1976-2000

Black cat, white cat, it doesn't matter as long as it catches mice—Deng Xiaoping, when asked about his plans to give private industry a greater role in China's economy

Whiskey's for drinkin' and water's for fightin' over—Mark Twain, on California water jurisdiction battles

The most controversial topic in statistics by far is that of **Bayesian** methods, the "California water" of the statistics world. In fact, it is so controversial that a strident Bayesian colleague of mine even took issue with my calling it "controversial"!

The name stems from Bayes' Rule (Section 2.6),

$$P(A|B) = \frac{P(A)P(B|A)}{P(A)P(B|A) + P(\text{not } A)P(B|\text{not } A)} \tag{22.92}$$

No one questions the validity of Bayes' Rule, and thus there is no controversy regarding statistical procedures that make use of probability calculations based on that rule. But the key word is *probability*. As long as the various terms in (22.92) are real probabilities—that is, based on actual data—there is no controversy.

But instead, the debate stems from the cases in which Bayesians replace some of the probabilities in the theorem with "feelings," i.e. non-probabilities, arising from what they call **subjective prior distributions**. The key word is then *subjective*. Our section here will concern the controversy over

the use of subjective priors.[9]

Say we wish to estimate a population mean. Here the Bayesian analyst, before even collecting data, says, "Well, I think the population mean could be 1.2, with probability, oh, let's say 0.28, but on the other hand, it might also be 0.88, with probability, well, I'll put it at 0.49..." etc. This is the analyst's subjective prior distribution for the population mean. The analyst does this before even collecting any data. Note carefully that he is NOT claiming these are real probabilities; he's just trying to quantify his hunches. The analyst then collects the data, and uses some mathematical procedure that combines these "feelings" with the actual data, and which then outputs an estimate of the population mean or other quantity of interest.

The Bayesians justify this by saying one should use all available information, even if it is just a hunch. "The analyst is typically an expert in the field under study. You wouldn't want to throw away his/her expertise, would you?" Moreover, they cite theoretical analyses that show that Bayes estimator doing very well in terms of criteria such as mean squared error, even if the priors are not "valid."

The non-Bayesians, known as **frequentists**, on the other hand dismiss this as unscientific and lacking in impartiality. "In research on a controversial health issue, say, you wouldn't want the researcher to incorporate his/her personal political biases into the number crunching, would you?" So, the frequentists' view is reminiscent of the Moynihan quoted above.

On the other hand, in the computer science world Bayesian estimation seems to be much less of a controversy. Computer scientists, being engineers, tend to be interested in whether a method seems to work, with the reasons being less important. This is the "black cat, white cat" approach.

By the way, the frequentists also point out that in the real world one must typically perform inference (confidence intervals or significance tests), not just compute point estimates; Bayesian methods are not really suited for inference.

Note carefully the key role of *data*. One might ask, for instance, "Why this sharp distinction between the Bayesians and the frequentists over the subjectivity issue? Don't the frequentists make subjective decisions too?" Consider an analysis of disk drive lifetime data, for instance. Some frequentist statistician might use a normal model, instead of, say, a gamma model. Isn't that subjectivity? The answer is no, because the statistician can *use the data* to assess the validity of her model, employing the methods of Section 26.2.

[9] By contrast, there is no controversy if the prior makes use of real data. I will explain this in Section 22.4.1.1 below, but in the mean time, note that my use of the term *Bayesian* refers only to subjective priors.

22.4.1 How It Works

To introduce the idea, consider again the example of estimating p, the probability of heads for a certain penny. Suppose we were to say—before tossing the penny even once—"I think p could be any number, but more likely near 0.5, something like a normal distribution with mean 0.5 and standard deviation, oh, let's say 0.1."

Of course, the true value of p is between 0 and 1, while the normal distribution extends from $-\infty$ to ∞. As noted in Section 8.16, the use of normal distributions is common for modeling bounded quantities like this one. Actually, many Bayesians prefer to use a beta distribution for the prior in this kind of setting, as the math works out more cleanly (derivations not shown here). But let's stick with the normal prior here for illustration.

The prior distribution is then $N(0.5, 0.1^2)$. But again, note that the Bayesians do not consider it to be a distribution in the sense of probability. It just quantifies our "gut feeling" here, our "hunch."

Nevertheless, in terms of the mathematics involved, it's as if the Bayesians are treating p as random, with p's distribution being whatever the analyst specifies as the prior. Under this "random p" assumption, the Maximum Likelihood Estimate (MLE), for instance, would change. Just as in the frequentist approach, the data here is X, the number of heads we get from n tosses of the penny. But in contrast to the frequentist approach, in which the likelihood would be

$$L = \left(\begin{array}{c} n \\ X \end{array} \right) p^X (1-p)^{n-X} \tag{22.93}$$

it now becomes

$$L = \frac{1}{\sqrt{2\pi}\ 0.1}\ \exp -0.5[(p-0.5)/0.1]^2 \left(\begin{array}{c} n \\ X \end{array} \right) p^X (1-p)^{n-X} \tag{22.94}$$

This is basically P(A and B) = P(A) P(B|A), though using a density rather than a probability mass function. We would then find the value of p which maximizes L, and take that as our estimate.

A Bayesian would use Bayes' Rule to compute the "distribution" of p given X, called the **posterior distribution**. The analog of (22.92) would be (22.94) divided by the integral of (22.94) as p ranges from 0 to 1, with the resulting quotient then being treated as a density. The (conditional) MLE would then be the **mode**, i.e. the point of maximal density of the posterior distribution.

But we could use any measure of central tendency, and in fact typically the mean is used, rather than the mode. In other words:

> To estimate a population value θ, the Bayesian constructs a prior "distribution" for θ (again, the quotation marks indicate that it is just a quantified gut feeling, rather

than a real probability distribution). Then she uses the prior together with the actual observed data to construct the posterior distribution. Finally, she takes her estimate $\hat{\theta}$ to be the mean of the posterior distribution.

Note how this procedure achieves a kind of balance between what our hunch says and what our data say. In (22.94), suppose the mean of p is 0.5 but n = 20 and X = 12 Then the frequentist estimator would be X/n = 0.6, while the Bayes estimator would be about 0.56. (Computation not shown here.) So our Bayesian approach "pulled" our estimate away from the frequentist estimate, toward our hunch that p is at or very near 0.5. This pulling effect would be stronger for smaller n or for a smaller standard deviation of the prior "distribution."

22.4.1.1 Empirical Bayes Methods

Note carefully that if the prior distribution in our model is not subjective, but is a real distribution verifiable from data, the above analysis on p would not be controversial at all. Say p does vary a substantial amount from one penny to another, so that there is a physical distribution involved. Suppose we have a sample of many pennies, tossing each one n times. If n is very large, we'll get a pretty accurate estimate of the value of p for each coin, and we can then plot these values in a histogram and compare it to the $N(0.5, 0.1^2)$ density, to check whether our prior is reasonable. This is called an **empirical Bayes** model, because we can empirically estimate our prior distribution, and check its validity. In spite of the name, frequentists would not consider this to be "Bayesian" analysis. Note that we could also assume that p has a general $N(\mu, \sigma^2)$ distribution, and estimate μ and σ from the data.

22.4.2 Extent of Usage of Subjective Priors

Though many statisticians, especially academics, are staunch, often militantly proselytizing, Bayesians, only a small minority of statisticians use the Bayesian approach **in practice**.

One way to see that Bayesian methodology is not mainstream is through the R programming language. For example, as of December 2010, only about 65 of the more than 3000 packages on CRAN, the R repository, involve Bayesian techniques. (See `http://cran.r-project.org/web/packages/tgp/index.html`.) There is actually a book on the topic, *Bayesian Computation with R*, by Jim Albert, Springer, 2007, and among those who use Bayesian techniques, many use R for that purpose. However, almost all general-purpose books on R do not cover Bayesian methodology at all.

Significantly, even among Bayesian academics, many use frequentist methods when they work on real, practical problems. Choose a Bayesian academic statistician at random, and you'll likely find on the Web that he/she does not use Bayesian methods when working on real applications.

On the other hand, use of subjective priors has become very common in the computer science research community. Papers using Bayesian methods appear frequently (no pun intended) in the CS research literature, and "seldom is heard a discouraging word."

22.4.3 Arguments Against Use of Subjective Priors

As noted, most professional statisticians are frequentists. In this section we will look at the arguments they have against the subjective Bayesian approach.

First, it's vital to reiterate a point made earlier:

> Frequentists have no objection at all to use of prior distributions based on actual data. They only object to use of subjective priors.

So, what is it about subjective priors that frequentists don't like?

The first point is that ultimately, the use of any statistical analysis is to make a decision about something. This could be a very formal decision, such as occurs when the Food and Drug Administration (FDA) decides whether to approve a new drug, or it could be informal, for instance when an ordinary citizen reads a newspaper article reporting on a study analyzing data on traffic accidents, and she decides what to conclude from the study.

There is nothing wrong using one's gut feelings to make a final decision, but it should not be part of the mathematical analysis of the data. One's hunches can play a role in deciding the "preponderance of evidence," as discussed in Section 21.11.5, but that should be kept separate from our data analysis.

If for example the FDA's data shows the new drug to be effective, but at the same time the FDA scientists still have their doubts, they may decide to delay approval of the drug pending further study. So they can certainly act on their hunch, or on non-data information they have, concerning approval of the drug. But the FDA, as a public agency, has a responsibility to the citizenry to state what the data say, i.e. to report the frequentist estimate, rather than merely reporting a number—the Bayesian estimate—that mixes fact and hunch.

In many if not most applications of statistics, there is a need for impartial estimates. As noted above, even if the FDA acts on a hunch to delay approval of a drug in spite of favorable data, the FDA owes the public (and the pharmaceutal firm) an impartial report of what the data say. Bayesian estimation is by definition not impartial. One Bayesian statistician friend put it very well, saying "I believe my own subjective priors, but I don't believe those of other people."

Furthermore, in practice we are typically interested in inference, i.e. confidence intervals and significance tests, rather than just point estimation. We are sampling from populations, and want

to be able to legitimately make inferences about those populations. For instance, though one can derive a Bayesian 95% confidence interval for p for our coin, it really has very little meaning, and again is certainly not impartial.

Some Bayesians justify their approach by pointing to the problems of p-values. Yet most frequentists also dislike p-values, and certainly use confidence intervals instead. The Bayesians who say one should uses subjective priors instead of p-values are simply setting up a sraw man, the critics say.

A common Bayesian approach concerns lower and upper bounds. The analyt feels sure that the true population value θ is, say, between r and s. He then chooses a prior distribution on (r,s), say uniform. Putting aside the question of the meaning of the results that ensure from the particular choice of prior, critics may ask, "What if the frequentist estimate turns out to be less than r? It could be a sampling artificat, but wouldn't you want to know about it, rather than automatically preventing such a situation?" This leads to the next section.

22.4.4 What Would You Do? A Possible Resolution

Consider the following scenario. Steven is running for president. Leo, his campaign manager, has commissioned Lynn to conduct a poll to assess Steven's current support among the voters. Lynn takes her poll, and finds that 57% of those polled support Steven. But her own gut feeling as an expert in politics, is that Steven's support is only 48%. She then combines these two numbers in some Bayesian fashion, and comes up with 50.2% as her estimate of Steven's support.

So, here the frequentist estimate is 57%, while Lynn's Bayesian estimate is 50.2%.

Lynn then gives Steven only the 50.2% figure, not reporting the value 57% number to him. Leo asks Lynn how she arrived at that number, and she explains that she combined her prior distribution with the data.

If you were Leo, what would you do? Consider two choices as to instructions you might give Lynn:

(a) You could say, "Lynn, I trust your judgment, so as the election campaign progresses, always give me only your Bayesian estimate."

(b) You might say, "Lynn, I trust your judgment, but as the election campaign progresses, always give me both your Bayesian estimate and what the impartial data actually say."

I believe that choice (b) is something that both the Bayesian and frequentist camps would generally agree upon.

22.4.5 The Markov Chain Monte Carlo Method

The computation of posterior distributions can involve complex multiple integrals. One could use numerical integration techniques, but again this can get complicated.

The *Markov Chain Monte Carlo* method approaches this problem by defining a certain Markov chain through the integration space, and then simulating that chain. The details are beyond the scope of this book, but here is yet another application of Markov chains.

22.4.6 Further Reading

Two UCD professors, the first current and the second former, have written interesting books about the Bayesian approach:

- *A Comparison of the Bayesian and Frequentist Approaches to Estimation*, Frank Samaniego, Springer, 2010.
- *Bayesian Ideas and Data Analysis: An Introduction for Scientists and Statisticians*, Chapman & Hall, 2010.

Exercises

1. Consider raffle ticket example in Section 22.1.1. Suppose 500 tickets are sold, and you have data on 8 of them. Continue to assume sampling with replacement. Consider the Maximum Likelihood and Methods of Moments estimators.

(a) Find the probability that the MLE is exactly equal to the true value of c.

(b) Find the exact probability that the MLE is within 50 of the true value.

(c) Find the approximate probability that the Method of Moments estimator is within 50 of the true value.

2. Suppose I = 1 or 0, with probability p and 1-p, respectively. Given I, X has a Poisson distribution with mean λ_I. Suppose we have $X_1, ..., X_n$, a random sample of size n from the (unconditional) distribution of X. (We do not know the associated values of I, i.e. $I_1, ..., I_n$.) This kind of situation occurs in various applications. The key point is the effect of the unseen variable. In terms of estimation, note that there are three parameters to be estimated.

(a) Set up the likelihood function, which if maximized with respect to the three parameters would yield the MLEs for them.

(b) The words *if* and *would* in that last sentence allude to the fact that MLEs cannot be derived in closed form. However, R's **mle()** function can be used to find their values numerically. Write R code to do this. In other words, write a function with a single argument **x**, representing the X_i, and returning the MLEs for the three parameters.

3. Find the Method of Moments and Maximum Likelihood estimators of the following parameters in famous distribution families:

- p in the binomial family (n known)
- p in the geometric family
- μ in the normal family (σ known)
- λ in the Poisson family

4. For each of the following quantities, state whether the given estimator is unbiased in the given context:

- $\hat{p}$, as an estimator of p, (20.5)
- $\hat{p}(1-\hat{p})$, as an estimator of p(1-p), (20.11)
- $\bar{X} - \bar{Y}$, as an estimator of $\mu_1 - \mu_2$, (20.6.1)
- $\frac{1}{n}\sum_{i=1}^{n}(X_i - \mu_1)^2$ (assuming μ_1 is known), as an estimator of σ_1^2, page 376
- $\bar{X}$, as an estimator of μ_1, page 376 *but sampling (from the population of Davis) without replacement*

5. Consider the Method of Moments Estimator $\hat{c}$ in the raffle example, Section 22.1.1. Find the exact value of $Var(\hat{c})$. Use the facts that $1 + 2 + ... + r = r(r+1)/2$ and $1^2 + 2^2 + ... + r^2 = r(r+1)(2r+1)/6$.

6. Suppose W has a uniform distribution on (-c,c), and we draw a random sample of size n, $W_1, ..., W_n$. Find the Method of Moments and Maximum Likelihood estimators. (Note that in the Method of Moments case, the first moment won't work.)

7. An urn contains ω marbles, one of which is black and the rest being white. We draw marbles from the urn one at a time, without replacement, until we draw the black one; let N denote the number of draws needed. Find the Method of Moments estimator of ω based on X.

8. Suppose $X_1, ..., X_n$ are uniformly distributed on (0,c). Find the Method of Moments and Maximum Likelihood estimators of c, and compare their mean squared error.

Hint: You will need the density of $M = \max_i X_i$. Derive this by noting that $M \leq t$ if and only if $X_i \leq t$ for all i = 1,2,...,n.

9. Add a single line to the code on page 365 that will print out the estimated value of Var(W).

10. In the raffle example, Section 22.1.1, find a $(1 - \alpha)\%$ confidence interval for c based on $\check{c}$, the Maximum Likelihood Estimate of c.

11. In many applications, observations come in correlated clusters. For instance, we may sample r trees at random, then s leaves within each tree. Clearly, leaves from the same tree will be more similar to each other than leaves on different trees.

In this context, suppose we have a random sample $X_1, ..., X_n$, n even, such that there is correlation within pairs. Specifically, suppose the pair (X_{2i+1}, X_{2i+2}) has a bivariate normal distribution with mean (μ, μ) and covariance matrix

$$\begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix} \tag{22.95}$$

i = 0,...,n/2-1, with the n/2 pairs being independent. Find the Method of Moments estimators of μ and ρ.

12. Suppose we have a random sample $X_1, ..., X_n$ from some population in which $EX = \mu$ and $Var(X) = \sigma^2$. Let $\overline{X} = (X_1 + ... + X_n)/n$ be the sample mean. Suppose the data points X_i are collected by a machine, and that due to a defect, the machine always records the last number as 0, i.e. $X_n = 0$. Each of the other X_i is distributed as the population, i.e. each has mean μ and variance σ^2. Find the mean squared error of $\overline{X}$ as an estimator of μ, separating the MSE into variance and squared bias components as in Section 22.2.

13. Suppose we have a random sample $X_1, ..., X_n$ from a population in which X is uniformly distributed on the region $(0, 1) \cup (2, c)$ for some unknown c > 2. Find closed-form expressions for the Method of Moments and Maximum Likelihood Estimators, to be denoted by T_1 and T_2, respectively.

Chapter 24

Mixture Models

Mixture distributions pop up in lots of places in probabilistic modeling, so we devote a chapter to them here.

24.1 The Old Trick Coin Example, Updated

Recall the trick coin example in Section 5.1. Let's look at a somewhat different version.

Suppose we have a large box of coins of two types. One type has probability p of heads, the other probability q of heads. A proportion r of the coins is of the first type. We reach into the box, choose a coin at random, and toss that coin 10 times, resulting in N heads. What is the distribution of N?

Since N is discrete, the distribution is described by its pmf, which is

$$p_N(k) = r\binom{10}{k}p^k(1-p)^{10-k} + (1-r)\binom{10}{k}q^k(1-q)^{10-k} \tag{24.1}$$

You can see why this is called a *mixture* model. The above pmf is a mixture of two binomial pmfs, with mixing proportions r and 1-r.

24.2 General Mixture Distributions

The general definition is rather abstract. Let's first restrict it to discrete outputs:

Definition 47 *Let $\{g_t\}_{t\in A}$ be a collection of pmfs, and let M be a random variable taking values*

in A. Consider discrete and continous cases for M, as follows. The **mixture distribution** *of the g_t with weights given by the distribution of M is defined as follows:*

- *M is discrete: The mixture distribution is the pmf*[1]

$$h = \sum_{i \in A} p_M(i)\ g_i \tag{24.2}$$

- *M is continuous: The mixture distribution has the pmf*

$$h(u) = \int_{t \in A} f_M(t)\ g_t(u)\ dt \tag{24.3}$$

For continuous outcomes, replace "pmf" by "density" everywhere above.

Actually, the definition isn't so abstract after all. A random variable Y having pmf or density h arises first of the random variable M, and then if M = i, getting Y randomly from the distribution g_i. A "notebook" analysis would have columns for both M and Y.

In other words, the pmf of Y is a weighted average of the g_i, with weights being the pmf of M. Note that this implies that the weights most be nonnegative and sum to 1.[2]

As noted, the above definition is pretty abstract, but it can be readily grasped by considering the above trick coin example. Here,

- $Y = N$
- A = {0,1}
- $p_M(0) = p_M(1) = 0.5$
- g_0 = the pmf for B(i,0.1)
- g_1 = the pmf for B(i,0.9)

[1] The reader should verify that H does qualify as a pmf, i.e. it consists of numbers in [0,1] that sum to 1.
[2] Some readers may recognize this as a **convex combination**.

Mixture distributions occur in a surprisingly wide variety of applications, and thus this material should be mastered. It is somewhat difficult at first, but if you always keep concrete examples such as the trick coin problem in mind, it is not bad at all.

By the way, there is nothing in the above definition that limits the G_i and H to be for scalar random variables. They could all be bivariate, for instance (though of course they all have to be of the same dimension.)

24.3 Generating Random Variates from a Mixture Distribution

One way to get additional insight on mixture distributions is to think about how to simulate them. For instance, consider the trick coin problem. The following R code simulates X_i:

```
rxi <- function(i) {
   m <- sample(0:1,1)
   p <- if (M==0) 0.1 else 0.9
   rbinom(1,i,p)
}
```

In general:

- We first generate M.
- Then we generate a random variable from the pmf G_M.

Various examples of mixture distributions will be presented in this chapter. But first, we need to develop some machinery.

24.4 A Useful Tool: the Law of Total Expectation

Let's put the mixture distribution issue aside for a moment, to discuss a tool that is useful both in mixture models and elsewhere.

The notion of iterated expectation, which was introduced in Sections 4.15 and 7.77, is an extremely important tool for the analysis of mixture distributions. We first need to extend our earlier definitions somewhat.

Again, this will seem rather abstract at first, but you'll see that it is a great convenience.

24.4.1 Conditional Expected Value As a Random Variable

For random variables Y and W, and an event, say W = t, the quantity E(Y|W = t) is the long-run average of Y, among the times when W = t occurs.

Note several things about the expression E(Y|W = t):

- The item to the left of the | symbol is a *random variable* (Y).
- The item on the right of the | symbol is an *event* (W = t).
- The overall expression evaluates to a constant.

By contrast, for the quantity E(Y|W) to be defined shortly, it is the case that:

- The item to the left of the | symbol is a random variable (Y).
- The item to the right of the | symbol is a random variable (W).
- The overall expression itself is a random variable, not a constant.

It will be very important to keep these differences in mind.

Consider the function g(t) defined as[3]

$$g(t) = E(Y|W = t) \tag{24.4}$$

In this case, the item to the right of the | is an event, and thus g(t) is a constant (for each value of t), not a random variable.

Definition 48 *Define g() as in (24.4). Form the new random variable Q = g(W). Then the quantity E(Y|W) is defined to be Q.*

(Before reading any further, re-read the two sets of bulleted items above, and make sure you understand the difference between E(Y|W=t) and E(Y|W).)

One can view E(Y|W) as a projection in an abstract vector space. This is very elegant, and actually aids the intuition. If (and only if) you are mathematically adventurous, read the details in Section 24.10.2.

[3] Of course, the t is just a placeholder, and any other letter could be used.

24.4.2 Famous Formula: Theorem of Total Expectation

An extremely useful formula, given only scant or no mention in most undergraduate probability courses, is

$$E(Y) = E[E(Y|W)] \tag{24.5}$$

for any random variables Y and W (for which the expectations are defined).

The RHS of (24.5) looks odd at first, but it's merely E[g(W)]; since Q = E(Y|W) is a random variable, we can certainly ask what its expected value is.

Equation (24.5) is a bit abstract. It's a very useful abstraction, enabling streamlined writing and thinking about the probabilistic structures at hand. Be sure to review the intuitive explanation following (4.68) before continuing.

24.4.3 Properties of Conditional Expectation and Variance

Conditional expected value is still expected value, and thus still has the usual properties of expected value. In particular,

$$E(aU + bV \mid W) = aE(U \mid W) + bE(V \mid W) \tag{24.6}$$

for any constants a and b. The same is true for variance, e.g.

$$Var(aU|W) = a^2 Var(U \mid W) \tag{24.7}$$

for any constants a, and

$$Var(U + V \mid W) = Var(U \mid W) + Var(V \mid W) \tag{24.8}$$

if U and V are *conditionally independent*, given W.

The reason for all this is simple. Consider the discrete case, for convenience. As seen in (3.13), E(Z) is a weighted average of the values of Z, with the weights being the values of p_Z:

$$EZ = \sum_c c p_Z(c) \tag{24.9}$$

But $E(Z|T)$ is simply another weighted average, with the weights being the conditional distribution of Z given T:

$$E(Z|T) = \sum_c c p_{Z|T}(c) \tag{24.10}$$

Note that (continuing the discrete case)

$$p_{Z|T=r}(c) = \frac{P(Z = c, T = r)}{P(T = r)} \tag{24.11}$$

All this extends to variance, since it too is defined in terms of expected value, in (3.38). So,

$$Var(Z|T) = E\{[Z - E(Z|T)]^2\} \tag{24.12}$$

And of course, if the variables above are continuous, just replace pmfs for densities, and sums by integrals.

24.4.4 Example: More on Flipping Coins with Bonuses

Recall the situation of Section 4.12: A game involves flipping a coin k times. Each time you get a head, you get a bonus flip, not counted among the k. (But if you get a head from a bonus flip, that does not give you its own bonus flip.) Let Y denote the number of heads you get among all flips, bonus or not. We'll compute EY.

Let

$$W = \text{number of heads from nonbonus flips} \tag{24.13}$$
$$= \text{number of bonus flips} \tag{24.14}$$

This is a natural situation in which to try the Theorem of Total Expectation, conditioning on Y. Reason as follows:

Y-W is the number of heads obtained from the bonus flips. Given W, the random variable Y-W has a binomial distribution with parameters W and 0.5, so

$$E(Y - W \mid W) = 0.5W \tag{24.15}$$

Then from (24.5),

$$\begin{aligned} EY &= E[E(Y \mid W)] && (24.16)\\ &= E\left[E(\{Y-W\}+W \mid W)\right] && (24.17)\\ &= E\left[E(Y-W|W)+E(W \mid W)\right] && (24.18)\\ &= E\left[0.5W+W\right] && (24.19)\\ &= 1.5EW && (24.20)\\ &= 0.75k, && (24.21) \end{aligned}$$

that last result coming from the fact that W itself has a binomial distribution with k trials and success probability 0.5.

Now here is the point: We could have used (4.68) above. But our use of (24.5) really streamlined things. If we had invoked (4.68), we would have had a messy sum, with binomial pmf expressions and so on. By using (24.5), we avoided that, a big win.

24.4.5 Example: Trapped Miner

This rather whimsical example is adapted from *Stochastic Processes,* by Sheldon Ross, Wiley, 1996.

A miner is trapped in a mine, and has a choice of three doors. Though he doesn't realize it, if he chooses to exit the first door, it will take him to safety after 2 hours of travel. If he chooses the second one, it will lead back to the mine after 3 hours of travel. The third one leads back to the mine after 5 hours of travel. Suppose the doors look identical, and if he returns to the mine he does not remember which door(s) he tried earlier. What is the expected time until he reaches safety?

The answer turns out to be 10. This is no accident, as 2+3+5 = 10. This was discovered on an intuitive basis (shown below) by UCD grad student Ahmed Ahmedin. Here is how we can derive the answer, using (24.5):

Let Y be the time needed to reach safety, let N denote the total number of attempts the miner makes before escaping (including the successful attempt at the end), and let U_i denote the time spent traveling during the i^{th} attempt, i = 1,...,N. Then

$$\begin{aligned} EY &= E(U_1+...,+U_N) && (24.22)\\ &= E\left[E(U_1+...+U_N|N)\right] && (24.23) \end{aligned}$$

Note carefully that the expression $U_1+...+U_N$ not only has random terms, but also the *number* of

terms, N, is random. So, technically, we cannot use (3.21) or its extension to multiple terms. But by conditioning on N, we are back in the situation of a fixed number of terms, and can proceed.

Given N, each of $U_1, ..., U_{N-1}$ takes on the values 3 and 5, with probability 0.5 each, while U_N is the constant 2. Thus

$$E(U_1 + ..., +U_N|N) = (N-1)\frac{3+5}{2} + 2 = 4N - 2 \tag{24.24}$$

N has a geometric distribution with success probability p = 1/3, so EN = 3. Putting all this together, we have

$$\begin{aligned} EY &= E(U_1 + ..., +U_N) && (24.25)\\ &= E\left[E(U_1 + ..., +U_N|N)\right] && (24.26)\\ &= E(4N-2) = 10 && (24.27) \end{aligned}$$

If 2, 3 and 5 were replaced by a, b and c, the result would turn out to be a+b+c. Intuitively: It takes an average of 3 attempts to escape, with each attempt having mean time of (a+b+c)/3, so the mean time overall is a+b+c.

Here is a different derivation (the one in Ross' book):

Let W denote the number of the door chosen (1, 2 or 3) on the first try. Then let us consider what values E(Y|W) can have. If W = 1, then Y = 2, so

$$E(Y|W=1) = 2 \tag{24.28}$$

If W = 2, things are a bit more complicated. The miner will go on a 3-hour excursion, and then be back in his original situation, and thus have a further expected wait of EY, since "time starts over." In other words,

$$E(Y|W=2) = 3 + EY \tag{24.29}$$

Similarly,

$$E(Y|W=3) = 5 + EY \tag{24.30}$$

In summary, now considering the *random variable* E(Y|W), we have

$$Q = E(Y|W) = \begin{cases} 2, & w.p. \frac{1}{3} \\ 3 + EY, & w.p. \frac{1}{3} \\ 5 + EY, & w.p. \frac{1}{3} \end{cases} \tag{24.31}$$

where "w.p." means "with probability." So, using (24.5) we have

$$EY = EQ = 2 \times \frac{1}{3} + (3 + EY) \times \frac{1}{3} + (5 + EY) \times \frac{1}{3} = \frac{10}{3} + \frac{2}{3} EY \tag{24.32}$$

Equating the extreme left and extreme right ends of this series of equations, we can solve for EY, which we find to be 10.

It is left to the reader to see how this would change if we assume that the miner remembers which doors he has already hit.

24.4.6 Example: Analysis of Hash Tables

(Famous example, adapted from various sources.)

Consider a database table consisting of m cells, only some of which are currently occupied. Each time a new key must be inserted, it is used in a hash function to find an unoccupied cell. Since multiple keys can map to the same table cell, we may have to probe multiple times before finding an unoccupied cell.

We wish to find E(Y), where Y is the number of probes needed to insert a new key. One approach to doing so would be to condition on W, the number of currently occupied cells at the time we do a search. After finding E(Y|W), we can use the Theorem of Total Expectation to find EY. We will make two assumptions (to be discussed later):

(a) Given that W = k, each probe will collide with an existing cell with probability k/m, with successive probes being independent.

(b) W is uniformly distributed on the set 1,2,...,m, i.e. P(W = k) = 1/m for each k.

To calculate E(Y|W=k), we note that given W = k, then Y is the number of independent trials until a "success" is reached, where "success" means that our probe turns out to be to an unoccupied cell. This is a **geometric** distribution, i.e.

$$P(Y = r|W = k) = (\frac{k}{m})^{r-1} (1 - \frac{k}{m}) \tag{24.33}$$

The mean of this geometric distribution is, from (4.4),

$$\frac{1}{1-\frac{k}{m}} \tag{24.34}$$

Then

$$\begin{aligned} EY &= E[E(Y|W)] && (24.35)\\ &= \sum_{k=1}^{m-1} \frac{1}{m} E(Y|W=k) && (24.36)\\ &= \sum_{k=1}^{m-1} \frac{1}{m-k} && (24.37)\\ &= 1 + \frac{1}{2} + \frac{1}{3} + ... + \frac{1}{m-1} && (24.38)\\ &\approx \int_1^m \frac{1}{u} du && (24.39)\\ &= \ln(m) && (24.40) \end{aligned}$$

where the approximation is something you might remember from calculus (you can picture it by drawing rectangles to approximate the area under the curve.).

Now, what about our assumptions, (a) and (b)? The assumption in (a) of each cell having probability k/m should be reasonably accurate if k is much smaller than m, because hash functions tend to distribute probes uniformly, and the assumption of independence of successive probes is all right too, since it is very unlikely that we would hit the same cell twice. However, if k is not much smaller than m, the accuracy will suffer.

Assumption (b) is more subtle, with differing interpretations. For example, the model may concern one specific database, in which case the assumption may be questionable. Presumably W grows over time, in which case the assumption would make no sense—it doesn't even *have* a distribution. We could instead think of a database which grows and shrinks as time progresses. However, even here, it would seem that W would probably oscillate around some value like m/2, rather than being uniformly distributed as assumed here. Thus, this model is probably not very realistic. However, even idealized models can sometimes provide important insights.

24.4.7 What About the Variance?

By the way, one might guess that the analog of the Theorem of Total Expectation for variance is

$$Var(Y) = E[Var(Y|W)] \tag{24.41}$$

But this is false. Think for example of the extreme case in which Y = W. Then Var(Y|W) would be 0, but Var(Y) should be nonzero.

The correct formula, called the Law of Total Variance, is

$$Var(Y) = E[Var(Y|W)] + Var[E(Y|W)] \tag{24.42}$$

Deriving this formula is easy, by simply evaluating both sides of (24.42), and using the relation $Var(X) = E(X^2) - (EX)^2$. This exercise is left to the reader. See also Section 24.10.2.

24.5 The EM Algorithm

Now returning to our main topic of mixture models, let's discuss a very common tool for fitting such models.

Consider again the example in Section 24.1. Now suppose p, q and r are all unknown, and we wish to estimate them from data. Say we will perform the above experiment 50 times, resulting in our data $N_1, ..., N_{50}$ We will estimate p, q and r by applying some method, say the Method of Moments, Maximum Likelihood or some ad hoc method of our own, to our data. Each N_i has the distribution seen in p_N above. This is a parametric family, with 3 parameters. (If, say, r is known, we only have a two-parameter family, and things are easier.)

So, how can we estimate those three parameters? In Chapter 22, we found some general methods for parameter estimation. Let's look at one of them, Maximum Likelihood Estimators (MLEs).

In (24.1), the MLE vector $(\widehat{p}, \widehat{q}, \widehat{r})'$ would be found by maximizing

$$\Pi_{i=1}^{n} \left[r \binom{10}{N_i} p^{N_i} (1-p)^{10-N_i} + (1-r) \binom{10}{N_i} q^{N_i} (1-q)^{10-N_i} \right] \tag{24.43}$$

After taking derivatives, etc., one would end up with a messy, nonlinear set of equations to solve. One could try **mle()**, but there is a better way to get the MLE: the Expection/Maximization (EM) algorithm. The derivation and ultimate formulas can get quite complex. Fortunately, R libraries

exist, such as **mixtools**, so you can avoid knowing all the details, as long as you understand the basic notion of a mixture model.

24.5.1 Overall Idea

In something like (24.2), for instance, one would make initial guesses for the $p_M(i)$ and then estimate the parameters of the g_i. In the next step, we'd do the opposite—take our current guesses for the latter parameters as known, and estimate the $p_M(i)$. Keep going until convergence.

To make things concrete, recall the trick coin example Section 24.1. But change it a little, so that the probabilities of heads for the two coins are unknown; call them p_0 (heads-light coin) and p_1 (heads-heavy coin). And also suppose that the two coins are not equally likely to be chosen, so that $p_M()$ is not known; denote P(M = 1) by q.

Suppose we have sample data, consisting of doing this experiment multiple times, say by reaching into the box n times and then doing m flips each time. We then wish to estimate 3 quantities—q and the two p_i—using our sample data.

We do so using the following iterative process. (The account here is not exactly EM, but captures the spirit of it.) We set up initial guesses, and iterate until convergence:

- **E step:** Update guess for q (complicated Bayes Rule equations).
- **M step:** Using the new guess for q, update the gueses for the two p_i.

The details are beyond the scope of this book.[4]

24.5.2 The mixtools Package

R's CRAN repository of contributed software includes the **mixtools** library. Let's see how to use one of the functions in that library, **normalmixEM()**, which assumes that the densities in (24.3) are from the normal distribution family.

Let's suppose we are modeling the data as a mixture of 2 normals. So, with our observed data set, our goal is to estimate 5 parameters from our data:

- μ_1, the mean of the first normal distribution
- μ_2, the mean of the second normal distribution

[4] "M" in the M step refers to the Maximum Likelihood method, a special case of the material in Section 22.1.3.

- σ_1, the mean of the second normal distribution
- σ_2, the mean of the second normal distribution
- q = P(M = 1) = 1 - P(M = 2)

The basic form of the call is

```
normalmixEM(x, lambda=NULL, mu=NULL, sigma=NULL, k=2)
```

where

- **x** is the data, a vector
- **lambda** is a vector of our initial guesses for the quantities P(M = i), of length **k** (recycling will be used if it is shorter); note that these must be probabilities summing to 1
- **mu** is a vector of our initial guesses for the μ_i
- **sigma** is a vector of our initial guesses for the σ_i
- **k** is the number of values M can take on, e.g. 2 for a mixture of 2 normals

One can leave the initial guesses as the default NULL values, but reasonable guesses may help convergence.

The return value will be an R list, whose components include **lambda**, **mu** and **sigma**, the final estimates of those values.

24.5.3 Example: Old Faithful Geyser

This example is taken from the online documentation in **mixtools**. The data concern the Old Faithful Geyser, a built-in data set in R.

The data here consist of waiting times between eruptions. A histogram, obtained via

```
> hist(faithful$waiting)
```

and shown in Figure 24.1, seems to suggest that the waiting time distribution is a mixture of 2 normals.

I tried initial guesses of means and standard deviations from the appearance of the histogram, and used equal weights for my initial guesses in that aspect:

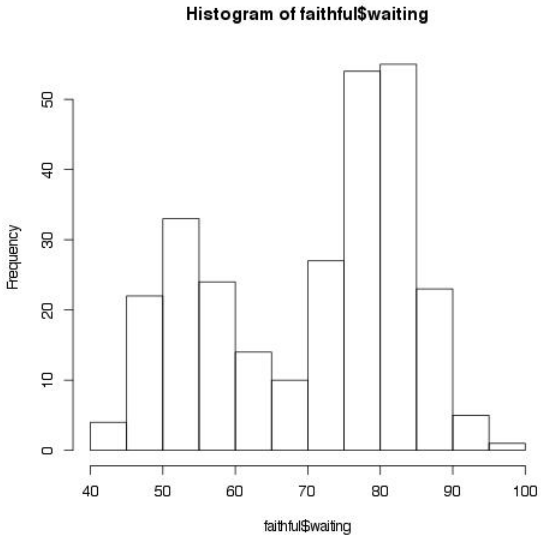

Figure 24.1: Old Faithful waiting times

```
> mixout <- normalmixEM(faithful$waiting,lambda=0.5,mu=c(55,80),sigma=10,k=2)
number of iterations= 9
> str(mixout)
List of 9
 $ x          : num [1:272] 79 54 74 62 85 55 88 85 51 85 ...
 $ lambda     : num [1:2] 0.361 0.639
 $ mu         : num [1:2] 54.6 80.1
 $ sigma      : num [1:2] 5.87 5.87
 $ loglik     : num -1034
 $ posterior  : num [1:272, 1:2] 1.02e-04 1.00 4.12e-03 9.67e-01 1.21e-06
...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:2] "comp.1" "comp.2"
 $ all.loglik: num [1:10] -1085 -1051 -1037 -1034 -1034 ...
 $ restarts   : num 0
 $ ft         : chr "normalmixEM"
 - attr(*, "class")= chr "mixEM"
```

So, the estimate from EM is that about 36% of the eruptions are of Type 1, etc. Interesting, when I tried it without my own initial guesses,

```
mixout <- normalmixEM(faithful$waiting,k=2)
```

the results were the same, so it seems to be a fairly stable model here.

By the way, since we are working with a hidden variable M here—in fact, we are merely postulating that it exists—how do we check this assumption? We'll return to this general idea of model fitting in Chapter 26.

24.6 Mean and Variance of Random Variables Having Mixture Distributions

Think of the random variables M and Y in the discussion following (24.3). Then EY is easy to find using the Law of Total Expectation:

$$EY = E[E(Y|M)] \tag{24.44}$$

Of course, evaluating this would require being able to compute E(Y | M), which is easy in some cases, not so easy in others.

Also, using the Law of Total Variance, we have that

$$Var(Y) = E[Var(Y|M)] + Var[E(Y|M)] \tag{24.45}$$

24.7 Example: Two Kinds of Batteries

Say a factory produces two kinds of batteries. One kind has lifetime that is exponentially distributed with mean 200 and the other's distribution is exponential with mean 500. Suppose 60% of the factory's production is of the former type, with 40% being of the latter type. Let's find the mean and variance of the lifetime Y of a randomly chosen battery.

Note that the distribution of Y is a mixture, in the sense of Section 24.2, in particular (24.3). Y here is a continuous random variable, referred to in that section as the "continuous outcome" case. In the notation there, let M be the battery type, with M being 0 or 1, for the 200-hour and 500-hour batteries, respectively. The conditional density of Y given M = 0 is exponential with mean 200, while given M = 1 it is exponential with mean 500. The unconditional density of Y is the mixture of these two exponential densities, as (24.3).

We want to find the unconditional mean and variance of Y.

Then

$$E(Y|M) = \begin{cases} 200, & w.p.\ 0.60 \\ 500, & w.p.\ 0.40 \end{cases} \tag{24.46}$$

and

$$Var(Y|M) = \begin{cases} 200^2, & w.p.\ 0.60 \\ 500^2, & w.p.\ 0.40 \end{cases} \tag{24.47}$$

(recalling that in the exponential family, variance is the square of the mean).

We can now use the formulas in Section 24.6. Let $Q_1 = E(Y|M)$ and $Q_2 = Var(Y|M)$. Then

$$EY = EQ_1 = 0.60 \times 200 + 0.40 \times 500 \tag{24.48}$$

and

$$\begin{aligned} Var(Y) &= E(Q_2) + Var(Q_1) && (24.49)\\ &= (0.60 \times 200^2 + 0.40 \times 500^2) + Var(Q_1) && (24.50)\\ &= (0.60 \times 200^2 + 0.40 \times 500^2) + E(Q_1^2) - (EQ_1)^2 && (24.51)\\ &= (0.60 \times 200^2 + 0.40 \times 500^2) + (0.60 \times 200^2 + 0.40 \times 500^2) && (24.52)\\ &\quad -(0.60 \times 200 + 0.40 \times 500)^2 && (24.53)\\ & && (24.54) \end{aligned}$$

24.8 Example: Overdispersion Models

A common model used in practice is that of **overdispersion**, in connection with Poisson models. Recall the following about the Poisson distribution family:

(a) This family is often used to model counts.

(b) For any Poisson distribution, the variance equals the mean.

In some cases in which we are modeling count data, condition (b) is too constraining. We want a "Poisson-ish" distribution in which the variance is greater than the mean, called **overdispersion**.

One may then try to fit a mixture of several Poisson distributions, instead of a single one. This does induce overdispersion, as we will now see.

Suppose M can equal 1,2,...,k, with probabilities $p_1, ..., p_k$ that sum to 1. Say the distribution of Y given M = i is Poisson with parameter λ_i. Then Y has a mixture distribution. Our goal here will be to show that Y is overdispersed, i.e. has a large variance than mean.

By the Law of Total Expectation,

$$\begin{aligned} EY &= E[E(Y|M)] && (24.55)\\ &= E(\lambda_M) && (24.56)\\ &= \sum_{i=1}^{k} p_i \lambda_i && (24.57) \end{aligned}$$

Note that in the above, the expression λ_M is a random variable, since its subscript M is random. Indeed, it is a function of M, so Equation (3.36) then applies, yielding the final equation. The random variable λ_M takes on the values $\lambda_1, ..., \lambda_k$ with probabilities $p_1, ..., p_k$, hence that final sum.

The corresponding formula for variance, (24.42), can be used to derive Var(Y).

$$\begin{aligned} Var(Y) &= E[Var(Y|M)] + Var[E(Y|M)] && (24.58)\\ &= E(\lambda_M) + Var(\lambda_M) && (24.59)\\ &= EY + Var(\lambda_M) && (24.60)\\ & && (24.61) \end{aligned}$$

Did you notice that this last equation achieves our goal of showing overdispersed? Since

$$Var(\lambda_M) > 0 \tag{24.62}$$

we have that

$$Var(Y) > E(\lambda_M) = EY \tag{24.63}$$

by (24.56).

But let's see just how much greater the variance is than the mean. The second term in (24.60) is evaluated the same way as in (24.56): This is the variance of a random variable that takes on the

values $\lambda_1, ..., \lambda_k$ with probabilities $p_1, ..., p_k$, which is

$$\sum_{i=1}^{k} p_i(\lambda_i - \overline{\lambda})^2 \tag{24.64}$$

where

$$\overline{\lambda} = E\lambda_M = \sum_{i=1}^{k} p_i \lambda_i \tag{24.65}$$

Thus

$$EY = \overline{\lambda} \tag{24.66}$$

and

$$Var(Y) = \overline{\lambda} + \sum_{i=1}^{k} p_i(\lambda_i - \overline{\lambda})^2 \tag{24.67}$$

So, as long as the λ_i are not equal, we have

$$Var(Y) > EY \tag{24.68}$$

in this Poisson mixture model, in contrast to the single-Poisson case in which Var(Y) = EY. You can now see why the Poisson mixture model is called an overdispersion model.

So, if one has count data in which the variance is greater than the mean, one might try using this model.

In mixing the Poissons, there is no need to restrict to discrete M. In fact, it is not hard to derive the fact that if X has a gamma distribution with parameters r and p/(1-p) for some $0 < p < 1$, and Y given X has a Poisson distribution with mean X, then the resulting Y neatly turns out to have a negative binomial distribution.

24.9 Example: Hidden Markov Models

The Hidden Markov Model to be introduced in Section 12.5 fits our definition of a mixture distribution, with M being the mixing variable.

According, the machinery of mixing models applies. For instance, we can use the EM algorithm to calculate various quantities.

24.10 Vector Space Interpretations (for the mathematically adventurous only)

The abstract vector space notion in linear algebra has many applications to statistics. We develop some of that material in this section.

Let $\mathcal{V}$ be the set of all such random variables having finite variance and mean 0. We can set up $\mathcal{V}$ as a vector space. For that, we need to define a sum and a scalar product. Define the sum of any two vectors X and Y to be the random variable X+Y. For any constant c, the vector cX is the random variable cX. Note that $\mathcal{V}$ is closed under these operations, as it must be: If X and Y both have mean 0, then X+Y does too, and so on.

Define an inner product on this space:

$$(X, Y) = Cov(X, Y) = E(XY) \tag{24.69}$$

(Recall that Cov(X,Y) = E(XY) - EX EY, and that we are working with random variables that have mean 0.) Thus the norm of a vector X is

$$||X|| = (X, X)^{0.5} = \sqrt{E(X^2)} = \sqrt{Var(X)} \tag{24.70}$$

again since E(X) = 0.

24.10.1 Properties of Correlation

The famous Cauchy-Schwarz Inequality for inner products says,

$$|(X, Y)| \leq ||X||\ ||Y|| \tag{24.71}$$

Applying that here, we have

$$|\rho(X, Y)| \leq 1 \tag{24.72}$$

So, vector space theory tells us that correlations are bounded between -1 and 1.

Also, the Cauchy-Schwarz Inequality yields equality if and only if one vector is a scalar multiple of the other, i.e. Y = cX for some c. When we then translate this to random variables of nonzero means, we get Y = cX + d.

In other words, the correlation between two random variables is between -1 and 1, with equality if and only if one is an exact linear function of the other.

24.10.2 Conditional Expectation As a Projection

For a random variable X in $\mathcal{V}$, let $\mathcal{W}$ denote the subspace of $\mathcal{V}$ consisting of all functions h(X) with mean 0 and finite variance. (Again, note that this subspace is indeed closed under vector addition and scalar multiplication.)

Now consider any Y in $\mathcal{V}$. Recall that the *projection* of Y onto $\mathcal{W}$ is the closest vector T in $\mathcal{W}$ to Y, i.e. T minimizes $||Y - T||$. That latter quantity is

$$\Big(E[(Y-T)^2]\Big)^{0.5} \tag{24.73}$$

To find the minimizing T, consider first the minimization of

$$E[(S-c)^2] \tag{24.74}$$

with respect to constant c for some random variable S. We already solved this problem back in Section 3.66. The minimizing value is c = ES.

Getting back to (24.73), use the Law of Total Expectation to write

$$E[(Y-T)^2] = E\Big(E[(Y-T)^2|X]\Big) \tag{24.75}$$

From what we learned with (24.74), applied to the conditional (i.e. inner) expectation in (24.75), we see that the T which minimizes (24.75) is T = E(Y|X).

In other words, the conditional mean is a projection! Nice, but is this useful in any way? The answer is yes, in the sense that it guides the intuition. All this is related to issues of statistical prediction—here we would be predicting Y from X—and the geometry here can really guide our insight. This is not very evident without getting deeply into the prediction issue, but let's explore some of the implications of the geometry.

For example, a projection is perpendicular to the line connecting the projection to the original

vector. So

$$0 = (E(Y|X), Y - E(Y|X)) = Cov[E(Y|X),\ Y - E(Y|X)] \tag{24.76}$$

This says that the prediction E(Y|X) is uncorrelated with the prediction error, Y-E(Y|X). This in turn has statistical importance. Of course, (24.76) could have been derived directly, but the geometry of the vector space intepretation is what suggested we look at the quantity in the first place. Again, the point is that the vector space view can guide our intuition.

Simlarly, the Pythagorean Theorem holds, so

$$||Y||^2 = ||E(Y|X)||^2 + ||Y - E(Y|X)||^2 \tag{24.77}$$

which means that

$$Var(Y) = Var[E(Y|X)] + Var[Y - E(Y|X)] \tag{24.78}$$

Equation (24.78) is a common theme in linear models in statistics, the decomposition of variance.

There is an equivalent form that is useful as well, derived as follows from the second term in (24.78). Since

$$E[Y - E(Y|X)] = EY - E[E(Y|X)] = EY - EY = 0 \tag{24.79}$$

we have

$$\begin{aligned} Var[Y - E(Y|X)] &= E\left[(Y - E(Y|X))^2\right] && (24.80)\\ &= E\left[Y^2 - 2Y E(Y|X) + (E(Y|X))^2\right] && (24.81) \end{aligned}$$

Now consider the middle term, $E[-2YE(Y|X)]$. Conditioning on X and using the Law of Total Expectation, we have

$$E[-2YE(Y|X)] = -2E\left[(E(Y|X))^2\right] \tag{24.82}$$

Then (24.80) becomes

$$
\begin{aligned}
Var[Y - E(Y|X)] &= E(Y^2) - E\left[(E(Y|X))^2\right] && (24.83)\\
&= E\left[E(Y^2|X)\right] - E\left[(E(Y|X))^2\right] && (24.84)\\
&= E\left(E(Y^2|X) - (E(Y|X))^2\right) && (24.85)\\
&= E\left[Var(Y|X)\right] && (24.86)
\end{aligned}
$$

the latter coming from our old friend, $Var(U) = E(U^2) - (EU)^2$, with U being Y here, under conditioning by X.

In other words, we have just derived another famous formula:

$$Var(Y) = E[Var(Y|X)] + Var[E(Y|X)] \tag{24.87}$$

24.11 Proof of the Law of Total Expectation

Let's prove (24.5) for the case in which W and Y take values only in the set {1,2,3,...}. Recall that if T is an integer-value random variable and we have some function h(), then L = h(T) is another random variable, and its expected value can be calculated as[5]

$$E(L) = \sum_k h(k) P(T = k) \tag{24.88}$$

In our case here, Q is a function of W, so we find its expectation from the distribution of W:

[5]This is sometimes called The Law of the Unconscious Statistician, by nasty probability theorists who look down on statisticians. Their point is that technically $EL = \sum_k kP(L = k)$, and that (24.88) must be proven, whereas the statisticians supposedly think it's a definition.

$$\begin{aligned}
E(Q) &= \sum_{i=1}^{\infty} g(i)P(W=i) \\
&= \sum_{i=1}^{\infty} E(Y|W=i)P(W=i) \\
&= \sum_{i=1}^{\infty} \left[\sum_{j=1}^{\infty} jP(Y=j|W=i) \right] P(W=i) \\
&= \sum_{j=1}^{\infty} j \sum_{i=1}^{\infty} P(Y=j|W=i)P(W=i) \\
&= \sum_{j=1}^{\infty} jP(Y=j) \\
&= E(Y)
\end{aligned}$$

In other words,

$$E(Y) = E[E(Y|W)] \tag{24.89}$$

Exercises

1. In the catchup game in Section 16.2.1, let V and W denote the winnings of the two players after only one turn. Find $P(V > 0.4)$.

2. Suppose one keeps rolling a die. Let S_n denote the total number of dots after n rolls, mod 8, and let T be the number of rolls needed for the event $S_n = 0$ to occur. Find $E(T)$, using an approach like that in the "trapped miner" example in Section 24.4.5.

3. In our ordinary coins which we use every day, each one has a slightly different probability of heads, which we'll call H. Say H has the distribution $N(0.5, 0.03^2)$. We choose a coin from a batch at random, then toss it 10 times. Let N be the number of heads we get. Find $Var(N)$.

4. Suppose the number N of bugs in a certain number of lines of code has a Poisson distribution, with parameter L, where L varies from one programmer to another. Show that Var(N) = EL + Var(L).

5. This problem arises from the analysis of random graphs, which for concreteness we will treat here as social networks such as Facebook.

In the model here, each vertex in the graph has N friends, N being a random variable with the same distribution at every vertex. One thinks of each vertex as generating its links, unterminated,

i.e. not tied yet to a second vertex. Then the unterminated links of a vertex pair off at random with those of other vertices. (Those that fail will just pair in self loops, but we'll ignore that.)

Let M denote the number of friends a friend of mine has. That is, start at a vertex A, and follow a link from A to another vertex, say B. M is the number of friends B has (we'll include A in this number).

(a) Since an unterminated link from A is more likely to pair up with a vertex that has a lot of links, a key assumption is that P(M = k) = ck P(N = k) for some constant c. Fill in the blank: This is an example of the setting we studied called _____.

(b) Show the following relation of generating functions: $g_M(s) = g'_N(s)/EN$.

6. Suppose Type 1 batteries have exponentially distributed lifetimes with mean 2.0 hours, while Type 2 battery lifetimes are exponentially distributed with mean 1.5. We have a large box containing a mixture of the two types of batteries, in proportions q and 1-q. We reach into the box, choose a battery at random, then use it. Let Y be the lifetime of the battery we choose. Use the Law of Total Variance, (24.42), to find $Var(Y)$.

7. In the backup battery example in Section 17.3.5, find Var(W), using the Law of Total Expectation.

8. Let X denote the number we obtain when we roll a single die once. Let $G_X(s)$ denote the generating function of X.

(a) Find $G_X(s)$.

(b) Suppose we roll the die 5 times, and let T denote the total number of dots we get from the 5 rolls. Find $G_T(s)$.

9. Consider this model of disk seeks. For simplicity, we'll assume a very tiny number of tracks, 3. Let X_1 and X_2 denote the track numbers of two successive disk requests. Each has a uniform distribution on {1,2,3}. But given $X_1 = i$, then $X_2 = i$ with probability 0.4, with X_2 being j with probability 0.3, $j \neq i$. (Convince yourself that these last two sentences are consistent with each other.) Find the following:

(a) $P(|X_1 - X_2| \leq 1)$

(b) $E(|X_1 - X_2|)$

(c) $F_{X_1,X_2}(2,2)$

10. Consider the computer worm example in Section 9.3.1. Let R denote the time it takes to go from state 1 to state 3. Find $f_R(v)$. (Leave your answer in integral form.)

11. Suppose (X,Y) has a bivariate normal distribution, with EX = EY = 0, Var(X) = Var(Y) = 1, and $\rho(X,Y) = 0.2$. Find the following, in integral forms:

(a) $E(X^2 + XY^{0.5})$

(b) $P(Y > 0.5X)$

(c) $F_{X,Y}(0.6, 0.2)$

12. Suppose X_i, i = 1,2,3,4,5 are independent and each have mean 0 and variance 1. Let $Y_i = X_{i+1} - X_i$, i = 1,2,3,4. Using the material in Section 16.4, find the covariance matrix of $Y = (Y_1, Y_2, Y_3, Y_4)$.

Chapter 25

Histograms and Beyond: Nonparametric Density Estimation

Here we will be concerned with estimating density functions in settings in which we do not assume our distribution belongs to some parametric model.

Why is this important? Actually, you've been seeing density estimates for years—except that they've been called *histograms*—and hopefully you are convinced that histograms are indeed useful tools for data visualization. Simply reporting the (estimated) mean and variance of a distribution may not capture the nuances.

Moreover, density estimates can be used in *clasification* problems, in which we are trying to get the class of something, based on other variables. In a medical context, for instance, we may be trying predict whether a patient will develop a certain disease, knowing her current test results, family history and so on. We'll cover such problems in Chapter 28, but for now what is important to know is that the clasification problem can be expressed in terms of ratios of densities.

But guess what! Histograms are actually density estimates, as we will see. And we can do better than histograms, with more sophisticated density estimates.

25.1 Example: Baseball Player Data

In the data in Section 20.9, suppose we are looking at player's weights W, and we are interested in $f_W(182)$, the population density for weights, evaluated at 182. Our data is considered to be a sample from the populatin of all major league players, past, present and future,[1] and we will use

[1] Recall Section 19.4.

this data to develop an estimate,

$$\widehat{f}_W(142) \tag{25.1}$$

for the population quantity

$$f_W(182) \tag{25.2}$$

But how can we do this?

25.2 Basic Ideas in Density Estimation

Suppose we have a random sample $R_1, ..., R_n$ from a distribution F_R. How can we estimate f_R from the R_i?

Recall from the first day of your calculus course that for a function $g()$

$$g'(x) = \lim_{h \to 0} \frac{g(x+h) - g(x)}{h} \tag{25.3}$$

and thus for small h,

$$g'(x) \approx \frac{g(x+h) - g(x)}{h} \tag{25.4}$$

Now take $g()$ to be $F_R()$ for our random variable R above, so that

$$f_R(t) \approx \frac{F_R(t+h) - F_R(t)}{h} \tag{25.5}$$

We can alter this a little: Instead of looking at the interval $(t, t+h)$, we can use $(t-h, t+h)$:

$$f_R(t) \approx \frac{F_R(t+h) - F_R(t-h)}{2h} \tag{25.6}$$

That means that

$$f_R(t) \approx \frac{F_R(t+h) - F_R(t-h)}{2h} \tag{25.7}$$
$$= \frac{P(R \leq t+h) - P(R \leq t-h)}{2h} \tag{25.8}$$
$$= \frac{P(t-h < R \leq t+h)}{2h} \tag{25.9}$$

if h is small. We can then form an estimate $\widehat{f_R}(t)$ by plugging in sample analogs in the right-hand side of (25.7):

$$\widehat{f_R}(t) = \frac{\#(t-h, t+h)/n}{2h} \tag{25.10}$$
$$= \frac{\#(t-h, t+h)}{2hn} \tag{25.11}$$

where the notation $\#(a, b)$ means the number of R_i in the interval (a,b).

There is an important issue of how to choose the value of h here, but let's postpone that for now. For the moment, let's take

$$h = \frac{\max_i R_i - \min_i R_i}{100} \tag{25.12}$$

i.e. take h to be 0.01 of the range of our data.

At this point, we'd then compute (25.11) at lots of different points t. Although it would seem that theoretically we must compute (25.11) at infinitely many such points, the graph of the function is actually a step function. Imagine t moving to the right, starting at $\min_i R_i$. The interval $(t-h, t+h)$ moves along with it. Whenever the interval moves enough to the right to either pick up a new R_i or lose one that it had had, (25.11) will change value, but not at any other time. So, we only need to evaluate the function at about $2n$ values of t.

25.3 Histograms

If for some reason we really want to save on computation, let's again say that we first break the interval $(\min_i R_i, \max_i R_i)$ into 100 subintervals of size h given by (25.12). We then compute (25.11) only at the midpoints of those intervals, and assume that the graph of $\widehat{f_R}(t)$ is approximately constant within each subinterval (true for small h). Do you know what we get from that? A

histogram! Yes, a histogram is a form of density estimation. (Usually a histogram merely displays counts. We do so here too, but we have scaled things so that the total area under the curve is 1, a property of densities.)

25.4 Kernel-Based Density Estimation

No matter what the interval width is, the histogram will consist of a bunch of rectangles, rather than a curve. We can get a smoother result if we used more sophisticated methods, one of which is called **kernel-based** density estimation. In base R, this is handled by the function **density()**.

For any particular value of t, $\widehat{f_R(t)}$ above depends only on the R_i that fall into that interval. If for instance some R_i is just barely outside the interval, it won't count. We'll now change that.

We need a set of weights, more precisely a weight function k, called the **kernel**. Any nonnegative function which integrates to 1—i.e. a density function in its own right—will work. Our estimator is then

$$\widehat{f_R}(t) = \frac{1}{nh} \sum_{i=1}^{n} k\left(\frac{t - R_i}{h}\right) \tag{25.13}$$

To make this idea concrete, take k to be the uniform density on (-1,1), which has the value 0.5 on (-1,1) and 0 elsewhere. Then (25.13) reduces to (25.11). Note how the parameter h, called the **bandwidth**, continues to control how far away from to t we wish to go for data points.

But as mentioned, what we really want is to include all data points, so we typically use a kernel with support on all of $(-\infty, \infty)$. In R's **density()** function, the default kernel is that of the N(0,1) density.

The bandwidth h controls how much smoothing we do; smaller values of h place heavier weights on data points near t and much lighter weights on the distant points.

There is no surefire way to choose a good bandwidth. A commonly used rule of thumb is

$$h = 1.06 \; s \; n^{-1/5} \tag{25.14}$$

where s is the sample standard deviation.

The default bandwidth in R is taken to the the standard deviation of k.

25.5 Example: Baseball Player Data

Some figures are plotted below for the baseball data, introduced in Section 20.9, for player weights, using functions in **ggplot2**:

- Figure 25.1 shows a histogram using the default number of bins, 30, programmed as follows

  ```
  p <- ggplot(baseball)
  p + geom_histogram(data=baseball, aes(x=Weight, y=..density..))
  ```

  As conceded in the documentation for **geom_histogram()**, the default tends to be not very good. This was the case here, with a very choppy figure.

- I then tried a binwidth of 10 pounds,

  ```
  p + geom_histogram(data=baseball, aes(x=Weight, y=..density..), binwidth=10)
  ```

  This gave the much smoother plot in Figure 25.2.

- I then tried a kernel density estimate with the default bandwidth:

  ```
  p + geom_density(aes(x=Weight))
  ```

  The result was similar to the histogram, but smoother, which is the goal.

- Finally, I superimposed on that last plot a plot for the catchers only (the latter in red):

  ```
  p + geom_density(aes(x=Weight)) +
      geom_density(data=catch, aes(x=Weight, colour="red"))
  ```

  As seen in Figure 25.4, the catchers tend to be a bit heavier, and have less variation than the players in general.

25.6 More on Density Estimation in ggplot2

See Section C.6.

25.7 Bias, Variance and Aliasing

Nonparametric density estimation gives us an opportunity to apply the principles of bias from Chapter 22.

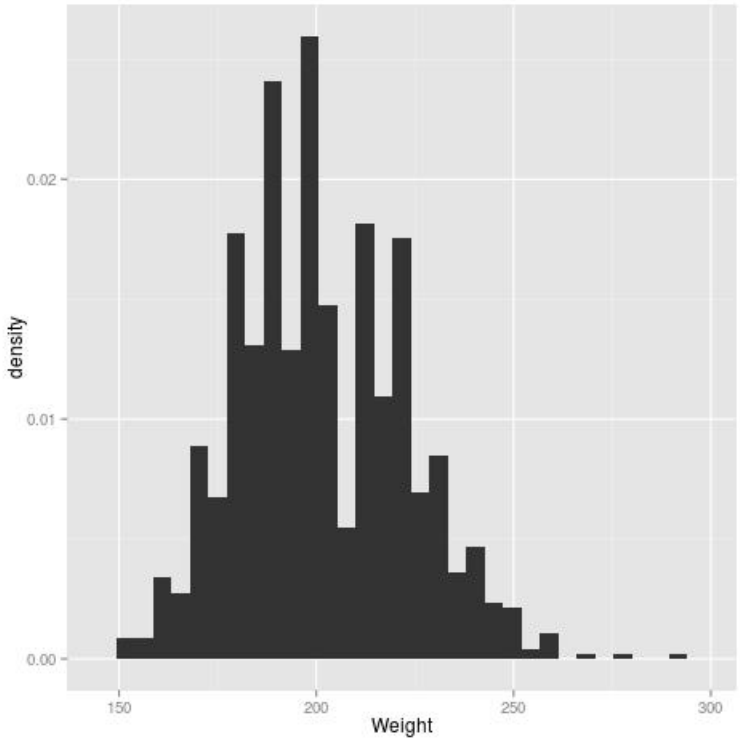

Figure 25.1: Histogram estimate, default binwidth

25.7.1 Bias vs. Variance

Recall from Section 22.2.5 that for an estimator $\widehat{\theta}$ of a population quantity θ we have that an overall measure of the accuracy of the estimator is

$$E[(\widehat{\theta} - \theta)^2] = bias(\widehat{\theta})^2 + Var(\widehat{\theta}) \tag{25.15}$$

In many cases there is a tradeoff between the bias and variance components here. We can have a smaller bias, for instance, but at the expense of increased variance. This is certainly the case with nonparametric density estimation.

As an illustration, suppose the true population density is $f_R(t) = 4t^3$ for t in (0,1), 0 elsewhere. Let's use (25.11):

$$\widehat{f}_R(t) = \frac{\#(t-h, t+h))}{2hn} \tag{25.16}$$

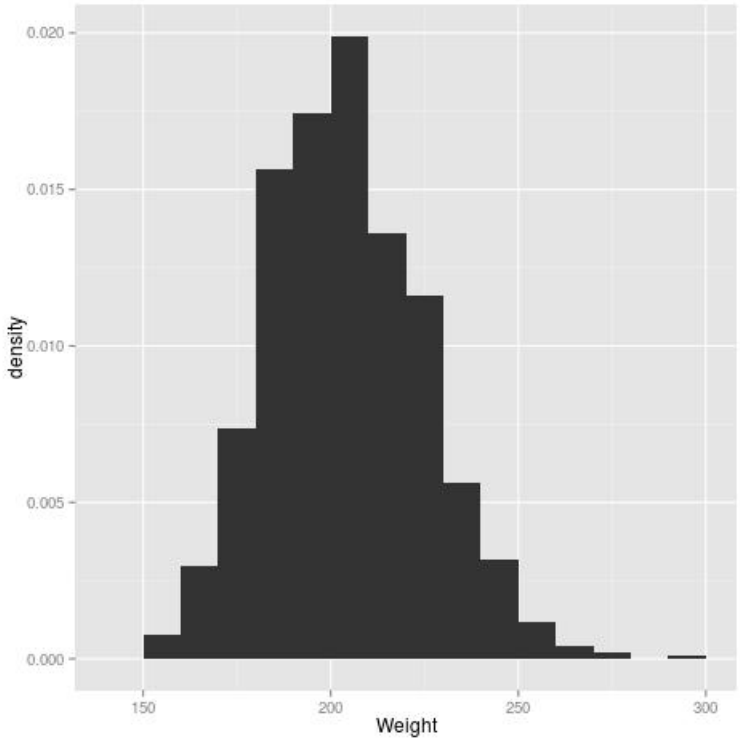

Figure 25.2: Histogram estimate, binwidth = 10

What is the bias? The numerator has a binomial distribution with n trials and success probability[2]

$$p = P(t-h < R < t+h) = \int_{t-h}^{t+h} 4u^3 \, du = (t+h)^4 - (t-h)^4 = 8t^3h + 8th^3 \tag{25.17}$$

By the binomial property, the numerator of (25.16) has expected value np, and thus

$$E[\widehat{f_R}(t)] = \frac{np}{2nh} = 4t^3 + 4th^2 \tag{25.18}$$

Subtracting $f_R(t)$, we have

$$bias[\widehat{f_R}(t)] = 4th^2 \tag{25.19}$$

[2]Note in the calculation here that it doesn't matter whether we write $\leq t+h$ or $< t+h$, since R is a continuous random variable.

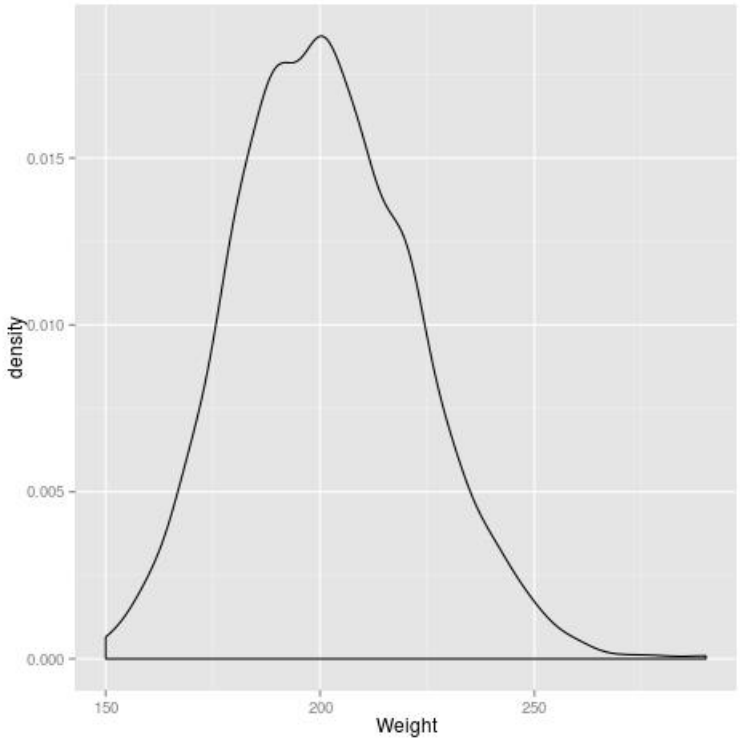

Figure 25.3: Kernel estimate, all players, default bandwidth

So, the smaller we set h, the smaller the bias, consistent with intuition.

How about the variance? Again using the binomial property, the variance of the numerator of (25.16) is np(1-p), so that

$$Var[\widehat{f_R}(t)] = \frac{np(1-p)}{(2nh)^2} = \frac{np}{2nh} \cdot \frac{1-p}{2nh} = (4t^3 + 4th^2) \cdot \frac{1-p}{2nh} \tag{25.20}$$

This matches intuition too: On the one hand, for fixed h, the larger n is, the smaller the variance of our estimator—i.e. larger samples are better, as expected. On the other hand, the smaller we set h, the larger the variance, because with small h there just won't be many R_i falling into our interval (t-h,t+h).

So, you can really see the bias-variance tradeoff here, in terms of what value we choose for h.[3]

[3]You might ask about finding the h to minimize (25.15). This would not make sense in our present context, in which we are simply assuming a known density in order to explore the bias and variance issues here. In practice, of

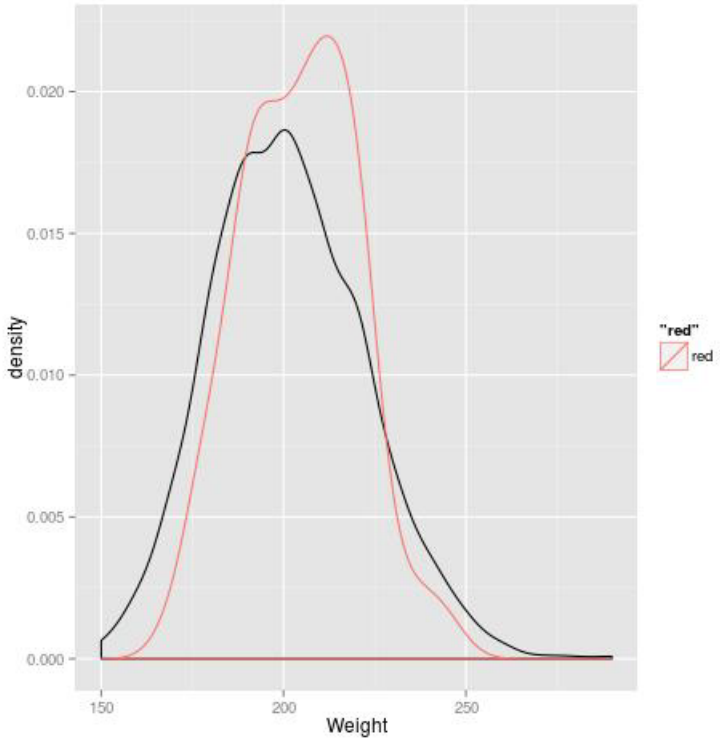

Figure 25.4: Kernel estimate, all players plus catchers (red), default bandwidth

25.7.2 Aliasing

There is another issue here to recognize: The integration in (25.17) tacitly assumed that $t - h > 0$ and $t + h < 1$. But suppose we are interested in $f_R(1)$. Then the upper limit in the integral in (25.17) will be 1, not t+h, which will approximately reduce the value of the integral by a factor of 2.

This results in strong bias near the endpoints of the support.[4] Let's illustrate this with the same density explored in our last section.

Using the general method in Section 7.11 for generating random numbers from a specified distribution, we have that this function will generate n random numbers from the density $4t^3$ on (0,1):

course, we don't know the density—that's why we are estimating it! However, some schemes ("plug-in" methods) for selecting h find a rough estimate of the density first, and then find the best h under the assumption that that estimate is correct.

[4]Recall that this term was defined in Section 7.5.

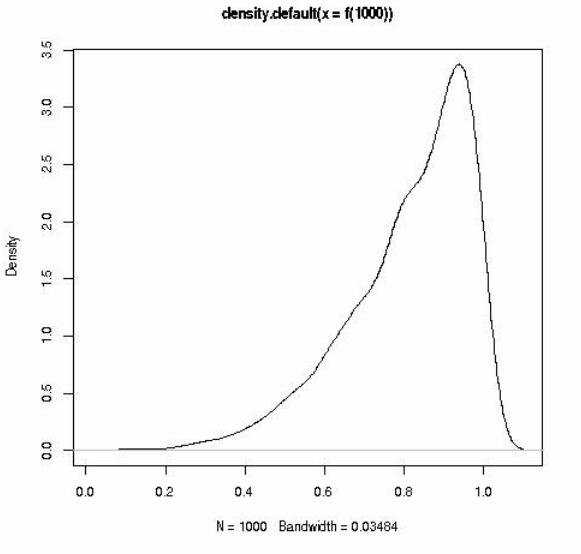

Figure 25.5: Example of Aliasing

```
f <- function(n) runif(n)^0.25
```

So, let's generate some numbers and plot the density estimate:

```
> plot(density(f(1000)))
```

The result is shown in Figure 25.5. Sure enough, the estimated density drops after about t = 0.9, instead of continuing to rise.

25.8 Nearest-Neighbor Methods

Consider (25.11) again. We count data points that are within a fixed distance from t; the number of such points will be random. With the nearest-neighbor approach, it's just the opposite: Now the number will be fixed, while the maximum distance from t will be random.

Specifically, at any point t we find the k nearest R_i to t, where k is chosen by the analyst just like h is selected in the kernel case. (And the method is are usually referred to as the *k-Nearest Neighbor* method, kNN.) The estimate is now

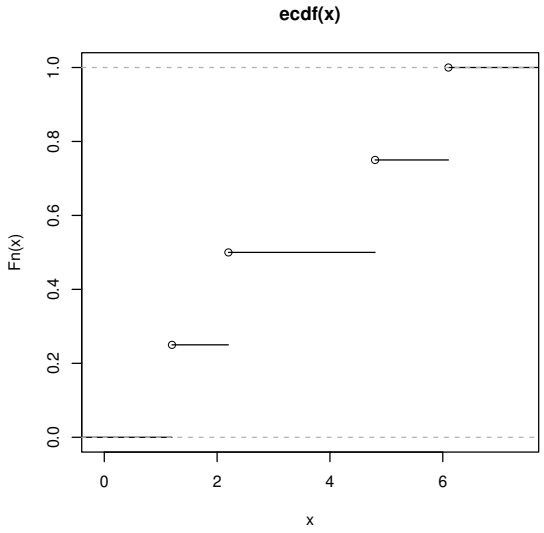

Figure 25.6: Empirical cdf, toy example

$$\widehat{f}_R(t) = \frac{k/n}{2 \max_i |R_i - t|} \tag{25.21}$$

$$= \frac{k}{2n \max_i |R_i - t|} \tag{25.22}$$

$$\tag{25.23}$$

25.9 Estimating a cdf

Let's introduce the notion of the **empirical distribution function** (ecdf), based on a sample $X_1, ..., X_n$. It is a sample estimate of a cdf, defined to be the proportion of X_i that are below t in the sample. Graphically, $\widehat{F}_X$ is a step function, with jumps at the values of the X_i.

As a small example, say n = 4 and our data are 4.8, 1.2, 2.2 and 6.1. We can plot the empirical cdf by calling R's **ecdf()** function:

```
> plot(ecdf(x))
```

The graph is in Figure 25.6. (In **ggplot2**, the function **stat_ecdf()** is similar.)

25.10 Hazard Function Estimation

In principle, estimating a hazard function from data should be a direct application of nonparametric density function methods. In (14.3) we would estimate the numerator with a kernel-based method, say, and the cdf in the denominator using the ecdf (Section 25.9).

However, the situation is complicated in that in many applications we have **censored data**, meaning that not all the data is available, due to an event not yet happening.

Say we measure lifetimes of batteries, and that at the time we collect our data, some of the batteries have died but some are still working. The problem is that we don't know the lifetimes of the latter. If one has run, say, for 212 hours so far, we know that it's lifetime will be at least 212, but we don't know the actual value yet.

This is an advanced topic, but a good starting point would be R's **muhaz** library in the CRAN repository. See the references in the documentation.

25.11 For Further Reading

To see an example of nonparametric density estimation applied to biology, see this paper by a UCD professor:

Kernel Methods in Line and Point Transect Sampling. *Biometrics*, Mack, Y. P. and P. X. Quang (1998). 54, 609-619.

Also see *All of Nonparametric Statistics*, Larry Wasserman Springer, 2007.

Appendix A

R Quick Start

Here we present a quick introduction to the R data/statistical programming language. Further learning resources are listed at `http://heather.cs.ucdavis.edu//r.html`.

R syntax is similar to that of C. It is object-oriented (in the sense of encapsulation, polymorphism and everything being an object) and is a functional language (i.e. almost no side effects, every action is a function call, etc.).

A.1 Correspondences

aspect	C/C++	R
assignment	=	<- (or =)
array terminology	array	vector, matrix, array
subscripts	start at 0	start at 1
array notation	m[2][3]	m[2,3]
2-D array storage	row-major order	column-major order
mixed container	struct, members accessed by .	list, members acessed by $ or [[ ]]
return mechanism	return	return() or last value computed
primitive types	int, float, double, char, bool	integer, float, double, character, logical
logical values	true, false	TRUE, FALSE (abbreviated T, F)
mechanism for combining modules	include, link	library()
run method	batch	interactive, batch
comment symbol	//	#

A.2 Starting R

To invoke R, just type "R" into a terminal window, e.g. **xterm** in Linux or Macs, **CMD** in Windows.

If you prefer to run from an IDE, you may wish to consider ESS for Emacs, StatET for Eclipse or RStudio, all open source. ESS is the favorite among the "hard core coder" types, while the colorful, easy-to-use, RStudio is a big general crowd pleaser. If you are already an Eclipse user, StatET will be just what you need.[1]

R is normally run in interactive mode, with > as the prompt. Among other things, that makes it easy to try little experiments to learn from; remember my slogan, "When in doubt, try it out!" For batch work, use **Rscript**, which is in the R package.

A.3 First Sample Programming Session

Below is a commented R session, to introduce the concepts. I had a text editor open in another window, constantly changing my code, then loading it via R's **source()** command. The original contents of the file **odd.R** were:

```
1 oddcount <- function(x)  {
2     k <- 0  # assign 0 to k
3     for (n in x)  {
4         if (n %% 2 == 1) k <- k+1  # %% is the modulo operator
5     }
6     return(k)
7 }
```

By the way, we could have written that last statement as simply

```
1     k
```

because the last computed value of an R function is returned automatically. This is actually preferred style in the R community.

The R session is shown below. You may wish to type it yourself as you go along, trying little experiments of your own along the way.[2]

[1] I personally use **vim**, as I want to have the same text editor no matter what kind of work I am doing. But I have my own macros to help with R work.

[2] The source code for this file is at `http://heather.cs.ucdavis.edu/~matloff/MiscPLN/R5MinIntro.tex`. You can download the file, and copy/paste the text from there.

```
> source("odd.R")  # load code from the given file
> ls()  # what objects do we have?
[1] "oddcount"
> # what kind of object is oddcount (well, we already know)?
> class(oddcount)
[1] "function"
> # while in interactive mode, and not inside a function, can print
> # any object by typing its name; otherwise use print(), e.g. print(x+y)
> oddcount  # a function is an object, so can print it
function(x)  {
   k <- 0  # assign 0 to k
   for (n in x)  {
      if (n %% 2 == 1) k <- k+1  # %% is the modulo operator
   }
   return(k)
}

> # let's test oddcount(), but look at some properties of vectors first
> y <- c(5,12,13,8,88)  # c() is the concatenate function
> y
[1]  5 12 13  8 88
> y[2]  # R subscripts begin at 1, not 0
[1] 12
> y[2:4]  # extract elements 2, 3 and 4 of y
[1] 12 13  8
> y[c(1,3:5)]  # elements 1, 3, 4 and 5
[1]  5 13  8 88
> oddcount(y)  # should report 2 odd numbers
[1] 2

> # change code (in the other window) to vectorize the count operation,
> # for much faster execution
> source("odd.R")
> oddcount
function(x)  {
   x1 <- (x %% 2 == 1)  # x1 now a vector of TRUEs and FALSEs
   x2 <- x[x1]  # x2 now has the elements of x that were TRUE in x1
   return(length(x2))
}

```

```
> # try it on subset of y, elements 2 through 3
> oddcount(y[2:3])
[1] 1
> # try it on subset of y, elements 2, 4 and 5
> oddcount(y[c(2,4,5)])
[1] 0

> # further compactify the code
> source("odd.R")
> oddcount
function(x)  {
   length(x[x %% 2 == 1])  # last value computed is auto returned
}
> oddcount(y)  # test it
[1] 2

# and even more compactification, making use of the fact that TRUE and
# FALSE are treated as 1 and 0
> oddcount <- function(x) sum(x %% 2 == 1)
# make sure you understand the steps that that involves:  x is a vector,
# and thus x %% 2 is a new vector, the result of applying the mod 2
# operation to every element of x; then x %% 2 == 1 applies the == 1
# operation to each element of that result, yielding a new vector of TRUE
# and FALSE values; sum() then adds them (as 1s and 0s)

# we can also determine which elements are odd
> which(y %% 2 == 1)
[1] 1 3
```

Note that the function of the R function **function()** is to produce functions! Thus assignment is used. For example, here is what **odd.R** looked like at the end of the above session:

```
oddcount <- function(x)  {
   x1 <- x[x %% 2 == 1]
   return(list(odds=x1, numodds=length(x1)))
}
```

We created some code, and then used **function()** to create a function object, which we assigned to **oddcount**.

A.4 Vectorization

Note that we eventually **vectorized** our function **oddcount()**. This means taking advantage of the vector-based, functional language nature of R, exploiting R's built-in functions instead of loops. This changes the venue from interpreted R to C level, with a potentially large increase in speed. For example:

```
> x <- runif(1000000)  # 1000000 random numbers from the interval (0,1)
> system.time(sum(x))
   user  system elapsed
  0.008   0.000   0.006
> system.time({s <- 0; for (i in 1:1000000) s <- s + x[i]})
   user  system elapsed
  2.776   0.004   2.859
```

A.5 Second Sample Programming Session

A matrix is a special case of a vector, with added class attributes, the numbers of rows and columns.

```
> # "rowbind() function combines rows of matrices; there's a cbind() too
> m1 <- rbind(1:2,c(5,8))
> m1
     [,1] [,2]
[1,]    1    2
[2,]    5    8
> rbind(m1,c(6,-1))
     [,1] [,2]
[1,]    1    2
[2,]    5    8
[3,]    6   -1

> # form matrix from 1,2,3,4,5,6, in 2 rows; R uses column-major storage
> m2 <- matrix(1:6,nrow=2)
> m2
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
> ncol(m2)
[1] 3
> nrow(m2)
```

```
[1] 2
> m2[2,3]  # extract element in row 2, col 3
[1] 6
# get submatrix of m2, cols 2 and 3, any row
> m3 <- m2[,2:3]
> m3
     [,1] [,2]
[1,]    3    5
[2,]    4    6

> m1 * m3  # elementwise multiplication
     [,1] [,2]
[1,]    3   10
[2,]   20   48
> 2.5 * m3  # scalar multiplication (but see below)
     [,1] [,2]
[1,]  7.5 12.5
[2,] 10.0 15.0
> m1 %*% m3  # linear algebra matrix multiplication
     [,1] [,2]
[1,]   11   17
[2,]   47   73

> # matrices are special cases of vectors, so can treat them as vectors
> sum(m1)
[1] 16
> ifelse(m2 %%3 == 1,0,m2) # (see below)
     [,1] [,2] [,3]
[1,]    0    3    5
[2,]    2    0    6
```

A.6 Recycling

The "scalar multiplication" above is not quite what you may think, even though the result may be. Here's why:

In R, scalars don't really exist; they are just one-element vectors. However, R usually uses **recycling**, i.e. replication, to make vector sizes match. In the example above in which we evaluated

the express `2.5 * m3`, the number 2.5 was recycled to the matrix

$$\begin{pmatrix} 2.5 & 2.5 \\ 2.5 & 2.5 \end{pmatrix} \tag{A.1}$$

in order to conform with **m3** for (elementwise) multiplication.

A.7 More on Vectorization

The **ifelse()** function is another example of vectorization. Its call has the form

```
ifelse(boolean vectorexpression1, vectorexpression2, vectorexpression3)
```

All three vector expressions must be the same length, though R will lengthen some via recycling. The action will be to return a vector of the same length (and if matrices are involved, then the result also has the same shape). Each element of the result will be set to its corresponding element in **vectorexpression2** or **vectorexpression3**, depending on whether the corresponding element in **vectorexpression1** is TRUE or FALSE.

In our example above,

```
> ifelse(m2 %%3 == 1,0,m2) # (see below)
```

the expression `m2 %%3 == 1` evaluated to the boolean matrix

$$\begin{pmatrix} T & F & F \\ F & T & F \end{pmatrix} \tag{A.2}$$

(TRUE and FALSE may be abbreviated to T and F.)

The 0 was recycled to the matrix

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{A.3}$$

while **vectorexpression3**, **m2**, evaluated to itself.

A.8 Third Sample Programming Session

This time, we focus on vectors and matrices.

```
> m <- rbind(1:3,c(5,12,13))  # "row bind," combine rows
> m
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    5   12   13
> t(m)  # transpose
     [,1] [,2]
[1,]    1    5
[2,]    2   12
[3,]    3   13
> ma <- m[,1:2]
> ma
     [,1] [,2]
[1,]    1    2
[2,]    5   12
> rep(1,2)  # "repeat," make multiple copies
[1] 1 1
> ma %*% rep(1,2)  # matrix multiply
     [,1]
[1,]    3
[2,]   17
> solve(ma,c(3,17))  # solve linear system
[1] 1 1
> solve(ma)  # matrix inverse
     [,1] [,2]
[1,]  6.0 -1.0
[2,] -2.5  0.5
```

A.9 Default Argument Values

Consider the **sort()** function, which is built-in to R, though the following points hold for any function, including ones you write yourself.

The online help for this function, invoked by

```
> ?sort
```

shows that the call form (the simplest version) is

```
sort(x, decreasing = FALSE, ...)
```

Here is an example:

```
> x <- c(12,5,13)
> sort(x)
[1]  5 12 13
> sort(x,decreasing=FALSE)
[1] 13 12  5
```

So, the default is to sort in ascending order, i.e. the argument **decreasing** has TRUE as its default value. If we want the default, we need not specify this argument. If we want a descending-order sort, we must say so.

A.10 The R List Type

The R **list** type is, after vectors, the most important R construct. A list is like a vector, except that the components are generally of mixed types.

A.10.1 The Basics

Here is example usage:

```
> g <- list(x = 4:6, s = "abc")
> g
$x
[1] 4 5 6

$s
[1] "abc"

> g$x  # can reference by component name
[1] 4 5 6
> g$s
[1] "abc"
> g[[1]]  # can reference by index, but note double brackets
[1] 4 5 6
> g[[2]]
[1] "abc"
> for (i in 1:length(g)) print(g[[i]])
[1] 4 5 6
[1] "abc"
```

```
# now have ftn oddcount() return odd count AND the odd numbers themselves,
# using the R list type
> source("odd.R")
> oddcount
function(x)  {
   x1 <- x[x %% 2 == 1]
   return(list(odds=x1, numodds=length(x1)))
}
> # R's list type can contain any type; components delineated by $
> oddcount(y)
$odds
[1]  5 13

$numodds
[1] 2

> ocy <- oddcount(y)  # save the output in ocy, which will be a list
> ocy
$odds
[1]  5 13

$numodds
[1] 2

> ocy$odds
[1]  5 13
> ocy[[1]]  # can get list elements using [[ ]] instead of $
[1]  5 13
> ocy[[2]]
[1] 2
```

A.10.2 The Reduce() Function

One often needs to combine elements of a list in some way. One approach to this is to use **Reduce()**:

```
> x <- list(4:6,c(1,6,8))
> x
[[1]]
[1] 4 5 6
```

```
[[2]]
[1] 1 6 8

> sum(x)
Error in sum(x) : invalid 'type' (list) of argument
> Reduce(sum,x)
[1] 30
```

Here **Reduce()** cumulatively applied R's **sum()** to **x**. Of course, you can use it with functions you write yourself too.

Continuing the above example:

```
> Reduce(c,x)
[1] 4 5 6 1 6 8
```

A.10.3 S3 Classes

R is an object-oriented (and functional) language. It features two types of classes, S3 and S4. I'll introduce S3 here.

An S3 object is simply a list, with a class name added as an *attribute*:

```
> j <- list(name="Joe", salary=55000, union=T)
> class(j) <- "employee"
> m <- list(name="Joe", salary=55000, union=F)
> class(m) <- "employee"
```

So now we have two objects of a class we've chosen to name **"employee"**. Note the quotation marks.

We can write class *generic functions*:

```
> print.employee <- function(wrkr) {
+     cat(wrkr$name,"\n")
+     cat("salary",wrkr$salary,"\n")
+     cat("union member",wrkr$union,"\n")
+ }
> print(j)
Joe
salary 55000
union member TRUE
```

```
> j
Joe
salary 55000
union member TRUE
```

What just happened? Well, **print()** in R is a *generic* function, meaning that it is just a placeholder for a function specific to a given class. When we printed **j** above, the R interpreter searched for a function **print.employee()**, which we had indeed created, and that is what was executed. Lacking this, R would have used the print function for R lists, as before:

```
> rm(print.employee)  # remove the function, to see what happens with print
> j
$name
[1] "Joe"

$salary
[1] 55000

$union
[1] TRUE

attr(,"class")
[1] "employee"
```

A.11 Some Workhorse Functions

```
> m <- matrix(sample(1:5,12,replace=TRUE),ncol=2)
> m
 [,1] [,2]
[1,]    2    1
[2,]    2    5
[3,]    5    4
[4,]    5    1
[5,]    2    1
[6,]    1    3
# call sum() on each row
> apply(m,1,sum)
[1] 3 7 9 6 3 4
# call sum() on each column
> apply(m,2,sum)
```

```
[1] 17 15
> f <- function(x) sum(x[x >= 4])
# call f() on each row
> apply(m,1,f)
[1] 0 5 9 5 0 0
> l <- list(x = 5, y = 12, z = 13)
# apply the given funciton to each element of l, producing a new list
> lapply(l,function(a) a+1)
$x
[1] 6

$y
[1] 13

$z
[1] 14
# group the first column of m by the second
> sout <- split(m[,1],m[,2])
> sout
$`1`
[1] 2 5 2
$`3`
[1] 1
$`4`
[1] 5
                                                                    $`5
[1] 2
# find the size of each group, by applying the length() function
> lapply(sout,length)
$`1`
[1] 3
                                                                    $`3
[1] 1
$`4`
[1] 1

$`5`
[1] 1
# like lapply(), but sapply() attempts to make vector output
> sapply(sout,length)
```

```
1 3 4 5
3 1 1 1
```

A.12 Handy Utilities

R functions written by others, e.g. in base R or in the CRAN repository for user-contributed code, often return values which are class objects. It is common, for instance, to have lists within lists. In many cases these objects are quite intricate, and not thoroughly documented. In order to explore the contents of an object—even one you write yourself—here are some handy utilities:

- **names()**: Returns the names of a list.
- **str()**: Shows the first few elements of each component.
- **summary()**: General function. The author of a class **x** can write a version specific to **x**, i.e. **summary.x()**, to print out the important parts; otherwise the default will print some bare-bones information.

For example:

```
> z <- list(a = runif(50), b = list(u=sample(1:100,25), v="blue sky"))
> z
$a
 [1] 0.301676229 0.679918518 0.208713522 0.510032893 0.405027042
0.412388038
 [7] 0.900498062 0.119936222 0.154996457 0.251126218 0.928304164
0.979945937
[13] 0.902377363 0.941813898 0.027964137 0.992137908 0.207571134
0.049504986
[19] 0.092011899 0.564024424 0.247162004 0.730086786 0.530251779
0.562163986
[25] 0.360718988 0.392522242 0.830468427 0.883086752 0.009853107
0.148819125
[31] 0.381143870 0.027740959 0.173798926 0.338813042 0.371025885
0.417984331
[37] 0.777219084 0.588650413 0.916212011 0.181104510 0.377617399
0.856198893
[43] 0.629269146 0.921698394 0.878412398 0.771662408 0.595483477
0.940457376
[49] 0.228829858 0.700500359
```

```
$b
$b$u
 [1] 33 67 32 76 29  3 42 54 97 41 57 87 36 92 81 31 78 12 85 73 26 44
86 40 43

$b$v
[1] "blue sky"
> names(z)
[1] "a" "b"
> str(z)
List of 2
 $ a: num [1:50] 0.302 0.68 0.209 0.51 0.405 ...
 $ b:List of 2
  ..$ u: int [1:25] 33 67 32 76 29 3 42 54 97 41 ...
  ..$ v: chr "blue sky"
> names(z$b)
[1] "u" "v"
> summary(z)
  Length Class  Mode
a 50     -none- numeric
b  2     -none- list
```

A.13 Data Frames

Another workhorse in R is the *data frame*. A data frame works in many ways like a matrix, but differs from a matrix in that it can mix data of different modes. One column may consist of integers, while another can consist of character strings and so on. Within a column, though, all elements must be of the same mode, and all columns must have the same length.

We might have a 4-column data frame on people, for instance, with columns for height, weight, age and name—3 numeric columns and 1 character string column.

Technically, a data frame is an R list, with one list element per column; each column is a vector. Thus columns can be referred to by name, using the **$** symbol as with all lists, or by column number, as with matrices. The matrix **a[i,j]** notation for the element of **a** in row **i**, column **j**, applies to data frames. So do the **rbind()** and **cbind()** functions, and various other matrix operations, such as filtering.

Here is an example using the dataset **airquality**, built in to R for illustration purposes. You can learn about the data through R's online help, i.e.

```
> ?airquality
```

Let's try a few operations:

```
> names(airquality)
[1] "Ozone"   "Solar.R" "Wind"    "Temp"    "Month"   "Day"
> head(airquality)  # look at the first few rows
  Ozone Solar.R Wind Temp Month Day
1    41     190  7.4   67     5   1
2    36     118  8.0   72     5   2
3    12     149 12.6   74     5   3
4    18     313 11.5   62     5   4
5    NA      NA 14.3   56     5   5
6    28      NA 14.9   66     5   6
> airquality[5,3]  # wind on the 5th day
[1] 14.3
> airquality$Wind[3]  # same
[1] 12.6
> nrow(airquality)  # number of days observed
[1] 153
> ncol(airquality)  # number of variables
[1] 6
> airquality$Celsius <- (5/9) * (airquality[,4] - 32)  # new variable
> names(airquality)
[1] "Ozone"   "Solar.R" "Wind"    "Temp"    "Month"   "Day"     "Celsius"
> ncol(airquality)
[1] 7
> airquality[1:3,]
  Ozone Solar.R Wind Temp Month Day  Celsius
1    41     190  7.4   67     5   1 19.44444
2    36     118  8.0   72     5   2 22.22222
3    12     149 12.6   74     5   3 23.33333
> aqjune <- airquality[airquality$Month == 6,]  # filter op
> nrow(aqjune)
[1] 30
> mean(aqjune$Temp)
[1] 79.1
> write.table(aqjune,"AQJune")  # write data frame to file
> aqj <- read.table("AQJune",header=T)  # read it in
```

A.14 Graphics

R excels at graphics, offering a rich set of capabilities, from beginning to advanced. In addition to the functions in base R, extensive graphics packages are available, such as **lattice** and **ggplot2**.

One point of confusion for beginners involves saving an R graph that is currently displayed on the screen to a file. Here is a function for this, which I include in my R startup file, **.Rprofile**, in my home directory:

```
pr2file
function (filename)
{
    origdev <- dev.cur()
    parts <- strsplit(filename, ".", fixed = TRUE)
    nparts <- length(parts[[1]])
    suff <- parts[[1]][nparts]
    if (suff == "pdf") {
        pdf(filename)
    }
    else if (suff == "png") {
        png(filename)
    }
    else jpeg(filename)
    devnum <- dev.cur()
    dev.set(origdev)
    dev.copy(which = devnum)
    dev.set(devnum)
    dev.off()
    dev.set(origdev)
}
```

The code, which I won't go into here, mostly involves manipulation of various R graphics devices. I've set it up so that you can save to a file of type either PDF, PNG or JPEG, implied by the file name you give.

A.15 Packages

The analog of a library in C/C++ in R is called a **package** (and often loosely referred to as a **library**). Some are already included in base R, while others can be downloaded, or written by yourself.

```
> library(parallel)  # load the package named 'parallel'
> ls(package:parallel)  # let's see what functions it gave us
 [1] "clusterApply"          "clusterApplyLB"        "clusterCall"
 [4] "clusterEvalQ"          "clusterExport"         "clusterMap"
 [7] "clusterSetRNGStream"   "clusterSplit"          "detectCores"
[10] "makeCluster"           "makeForkCluster"       "makePSOCKcluster"
[13] "mc.reset.stream"       "mcaffinity"            "mccollect"
[16] "mclapply"              "mcMap"                 "mcmapply"
[19] "mcparallel"            "nextRNGStream"         "nextRNGSubStream"
[22] "parApply"              "parCapply"             "parLapply"
[25] "parLapplyLB"           "parRapply"             "parSapply"
[28] "parSapplyLB"           "pvec"                  "setDefaultCluster"
[31] "splitIndices"          "stopCluster"
> ?pvec  # let's see how one of them works
```

The CRAN repository of contributed R code has thousands of R packages available. It also includes a number of "tables of contents" for specific areas, say time series, in the form of CRAN Task Views. See the R home page, or simply Google "CRAN Task View."

```
> install.packages("cts","~/myr")  # download into desired directory
--- Please select a CRAN mirror for use in this session ---
...
downloaded 533 Kb


The downloaded binary packages are in
        /var/folders/jk/dh9zkds97sj23kjcfkr5v6q00000gn/T//RtmplkKzOU/downloaded_pa
> ?library
> library(cts,lib.loc="~/myr")

Attaching package:      cts
...
```

A.16 Other Sources for Learning R

There are tons of resources for R on the Web. You may wish to start with the links at `http://heather.cs.ucdavis.edu/~matloff/r.html`.

A.17 Online Help

R's **help()** function, which can be invoked also with a question mark, gives short descriptions of the R functions. For example, typing

```
> ?rep
```

will give you a description of R's **rep()** function.

An especially nice feature of R is its **example()** function, which gives nice examples of whatever function you wish to query. For instance, typing

```
> example(wireframe())
```

will show examples—R code and resulting pictures—of **wireframe()**, one of R's 3-dimensional graphics functions.

A.18 Debugging in R

The internal debugging tool in R, **debug()**, is usable but rather primitive. Here are some alternatives:

- The RStudio IDE has a built-in debugging tool.
- For Emacs users, there is **ess-tracebug**.
- The StatET IDE for R on Eclipse has a nice debugging tool. Works on all major platforms, but can be tricky to install.
- My own debugging tool, **debugR**, is extensive and easy to install, but for the time being is limited to Linux, Mac and other Unix-family systems. See *http://heather.cs.ucdavis.edu/debugR.html*.

A.19 Complex Numbers

If you have need for complex numbers, R does handle them. Here is a sample of use of the main functions of interest:

```
> za <- complex(real=2,imaginary=3.5)
> za
```

```
[1] 2+3.5i
> zb <- complex(real=1,imaginary=-5)
> zb
[1] 1-5i
> za * zb
[1] 19.5-6.5i
> Re(za)
[1] 2
> Im(za)
[1] 3.5
> za^2
[1] -8.25+14i
> abs(za)
[1] 4.031129
> exp(complex(real=0,imaginary=pi/4))
[1] 0.7071068+0.7071068i
> cos(pi/4)
[1] 0.7071068
> sin(pi/4)
[1] 0.7071068
```

Note that operations with complex-valued vectors and matrices work as usual; there are no special complex functions.

A.20 Further Reading

For further information about R as a programming language, there is my book, *The Art of R Programming: a Tour of Statistical Software Design*, NSP, 2011, as well as Hadley Wickham's *Advanced R*, Chapman and Hall, 2014.

For R's statistical functions, a plethora of excellent books is available. such as *The R Book* (2nd Ed.), Michael Crowley, Wiley, 2012. I also very much like *R in a Nutshell* (2nd Ed.), Joseph Adler, O'Reilly, 2012, and even Andrie de Vries' *R for Dummies*, 2012.