4.3: Composite Trials

Composite trials and component events

Often a trial is a composite one. That is, the fundamental trial is completed by performing several steps. In some cases, the steps are carried out sequentially in time. In other situations, the order of performance plays no significant role. Some of the examples in the unit on Conditional Probability involve such multistep trials. We examine more systematically how to model composite trials in terms of events determined by the components of the trials. In the subsequent section, we illustrate this approach in the important special case of Bernoulli trials, in which each outcome results in a success or failure to achieve a specified condition.

We call the individual steps in the composite trial *component trials*. For example, in the experiment of flipping a coin ten times, we refer the ith toss as the ith component trial. In many cases, the component trials will be performed sequentially in time. But we may have an experiment in which ten coins are flipped simultaneously. For purposes of analysis, we impose an ordering— usually by assigning indices. The question is how to model these repetitions. Should they be considered as ten trials of a single simple experiment? It turns out that this is not a useful formulation. We need to consider the composite trial as a single outcome— i.e., represented by a single point in the basic space ω.

Some authors give considerable attention the the nature of the basic space, describing it as a Cartesian product space, with each coordinate corresponding to one of the component outcomes. We find that unnecessary, and often confusing, in setting up the basic model. We simply suppose the basic space has enough elements to consider each possible outcome. For the experiment of flipping a coin ten times, there must be at least $2^{10} = 1024$ elements, one for each possible sequence of heads and tails.

Of more importance is describing the various events associated with the experiment. We begin by identifying the appropriate *component events*. A component event is determined by propositions about the outcomes of the corresponding component trial.

Example 4.3.1 Component events

- In the coin flipping experiment, consider the event H_3 that the third toss results in a head. Each outcome ω of the experiment may be represented by a sequence of H's and T's, representing heads and tails. The event H_3 consists of those outcomes represented by sequences with H in the third position. Suppose A is the event of a head on the third toss and a tail on the ninth toss. This consists of those outcomes corresponding to sequences with H in the third position and T in the ninth. Note that this event is the intersection $H_3 H_9^c$.
- A somewhat more complex example is as follows. Suppose there are two boxes, each containing some red and some blue balls. The experiment consists of selecting at random a ball from the first box, placing it in the second box, then making a random selection from the modified contents of the second box. The composite trial is made up of two component selections. We may let R_1 be the event of selecting a red ball on the first component trial (from the first box), and R_2 be the event of selecting a red ball on the second component trial. Clearly R_1 and R_2 are component events.

In the first example, it is reasonable to assume that the class $\{H_i : 1 \le i \le 10\}$ is independent, and each component probability is usually taken to be 0.5. In the second case, the assignment of probabilities is somewhat more involved. For one thing, it is necessary to know the numbers of red and blue balls in each box before the composite trial begins. When these are known, the usual assumptions and the properties of conditional probability suffice to assign probabilities. This approach of utilizing component events is used tacitly in some of the examples in the unit on Conditional Probability.

When appropriate component events are determined, various Boolean combinations of these can be expressed as minterm expansions.

Example 4.3.2

Four persons take one shot each at a target. Let E_i be the event the ith shooter hits the target center. Let A_3 be the event exacty three hit the target. Then A_3 is the union of those minterms generated by the E_i which have three places uncomplemented.

$$A_3 = E_1 E_2 E_3 E_4^c \bigvee E_1 E_2 E_3^c E_4 \bigvee E_1 E_2^c E_3 E_4 \bigvee E_1^c E_2 E_3 E_4^c$$

Usually we would be able to assume the E_i form an independent class. If each $P(E_i)$ is known, then all minterm probabilities can be calculated easily.

The following is a somewhat more complicated example of this type.

Example 4.3.3

Ten race cars are involved in time trials to determine pole positions for an upcoming race. To qualify, they must post an average speed of 125 mph or more on a trial run. Let E_i be the event the ith car makes qualifying speed. It seems reasonable to suppose the class $\{E_i : 1 \le i \le 10\}$ is independent. If the respective probabilities for success are 0.90, 0.88, 0.93, 0.77, 0.85, 0.96, 0.72, 0.83, 0.91, 0.84, what is the probability that k or more will qualify ($k = 6, 7, 8, 9, 10$)?

Solution

Let A_k be the event exactly k qualify. The class $\{E_i : 1 \le i \le 10\}$ generates $2^{10} = 1024$ minterms. The event A_k is the union of those minterms which have exactly k places uncomplemented. The event B_k that k or more qualify is given by

$$B_k = \bigvee_{r=k}^{n} A_r$$

The task of computing and adding the minterm probabilities by hand would be tedious, to say the least. However, we may use the function ckn, introduced in the unit on MATLAB and Independent Classes and illustrated in Example 4.4.2, to determine the desired probabilities quickly and easily.

```
>> P = [0.90, 0.88, 0.93, 0.77, 0.85, 0.96,0.72, 0.83, 0.91, 0.84];
>> k = 6:10;
>> PB = ckn(P,k)
PB =   0.9938    0.9628    0.8472    0.5756    0.2114
```

An alternate approach is considered in the treatment of random variables.

Bernoulli trials and the binomial distribution

Many composite trials may be described as a sequence of *success-failure* trials. For each component trial in the sequence, the outcome is one of two kinds. One we designate a *success* and the other a *failure*. Examples abound: heads or tails in a sequence of coin flips, favor or disapprove of a proposition in a survey sample, and items from a production line meet or fail to meet specifications in a sequence of quality control checks. To represent the situation, we let E_i be the event of a success on the ith component trial in the sequence. The event of a failure on the ith component trial is thus E_i^c.

In many cases, we model the sequence as a *Bernoulli sequence*, in which the results on the successive component trials are independent and have the same probabilities. Thus, formally, a sequence of success-failure trials is Bernoulli iff

The class $\{E_i : 1 \le i\}$ is independent.
The probability $P(E_i) = p$, invariant with i.

Simulation of Bernoulli trials

It is frequently desirable to simulate Bernoulli trials. By flipping coins, rolling a die with various numbers of sides (as used in certain games), or using spinners, it is relatively easy to carry this out physically. However, if the number of trials is large—say several hundred—the process may be time consuming. Also, there are limitations on the values of p, the probability of success. We have a convenient two-part m-procedure for simulating Bernoulli sequences. The first part, called *btdata*, sets the parameters. The second, called bt, uses the random number generator in MATLAB to produce a sequence of zeros and ones (for failures and successes). Repeated calls for bt produce new sequences.

Example 4.3.4

```
>> btdata
Enter n, the number of trials  10
Enter p, the probability of success on each trial  0.37
 Call for bt
 >> bt
```

```
n = 10   p = 0.37     % n is kept small to save printout space
Frequency = 0.4
To view the sequence, call for SEQ
>> disp(SEQ)          % optional call for the sequence
     1     1
     2     1
     3     0
     4     0
     5     0
     6     0
     7     0
     8     0
     9     1
    10     1
```

Repeated calls for bt yield new sequences with the same parameters.

To illustrate the power of the program, it was used to take a run of 100,000 component trials, with probability p of success 0.37, as above. Successive runs gave relative frequencies 0.37001 and 0.36999. Unless the random number generator is "seeded" to make the same starting point each time, successive runs will give different sequences and usually different relative frequencies.

The binomial distribution

A basic problem in Bernoulli sequences is to determine the probability of k successes in n component trials. We let S_n be the number of successes in n trials. This is a special case of a simple random variable, which we study in more detail in the chapter on "Random Variables and Probabilities".

Let us characterize the events $A_{kn} = \{S_n = k\}$, $0 \le k \le n$. As noted above, the event A_{kn} of exactly k successes is the union of the minterms generated by $\{E_i : 1 \le i \}$ in which there are k successes (represented by k uncomplemented E_i) and $n - k$ failures (represented by $n - k$ complemented E_i^c). Simple combinatorics show there are $C(n, k)$ ways to choose the k places to be uncomplemented. Hence, among the 2^n minterms, there are $C(n, k) = \dfrac{n!}{k!(n-k)!}$ which have k places uncomplemented. Each such minterm has probability $p^k(1-p)^{n-k}$. Since the minterms are mutually exclusive, their probabilities add. We conclude that

$$P(S_n = k) = C(n, k)p^k(1-p)^{n-k} = C(n, k)p^k q^{n-k} \quad \text{where } q = 1 - p \text{ for } 0 \le k \le n$$

These probabilities and the corresponding values form the *distribution* for S_n. This distribution is known as the *binomial distribution*, with parameters (n, p). We shorten this to binomial (n, p), and often writ S_n ~ binomial (n, p). A related set of probabilities is $P(S_n \ge k) = P(B_{kn})$, $0 \le k \le n$. If the number n of component trials is small, direct computation of the probabilities is easy with hand calculators.

Example 4.3.5 A reliability problem

A remote device has five similar components which fail independently, with equal probabilities. The system remains operable if three or more of the components are operative. Suppose each unit remains active for one year with probability 0.8. What is the probability the system will remain operative for that long?

Solution

$$P = C(5, 3)0.8^3 \cdot 0.2^2 + C(5, 4)0.8^4 \cdot 0.2 + C(5, 5)0.8^5 = 10 \cdot 0.8^3 \cdot 0.2^2 + 5 \cdot 0.8^4 \cdot 0.2 + 0.8^5 = 0.9421$$

Because Bernoulli sequences are used in so many practical situations as models for success-failure trials, the probabilities $P(S_n = k)$ and $P(S_n \ge k)$ have been calculated and tabulated for a variety of combinations of the parameters (n, p). Such tables are found in most mathematical handbooks. Tables of $P(S_n = k)$ are usually given a title such as *binomial distribution, individual terms*. Tables of $P(S_n \ge k)$ have a designation such as *binomial distribution, cumulative terms*. Note, however, some tables for cumulative terms give $P(S_n \le k)$. Care should be taken to note which convention is used.

Example 4.3.6 A reliability problem

Consider again the system of Example 5, above. Suppose we attempt to enter a table of Cumulative Terms, Binomial Distribution at $n = 5$, $k = 3$, and $p = 0.8$. Most tables will *not* have probabilities greater than 0.5. In this case, we may work with *failures*. We just interchange the role of E_i and E_i^c. Thus, the number of failures has the binomial (n, p) distribution. Now there are three or more successes iff there are *not* three or more failures. We go the the table of cumulative terms at $n = 5$, $k = 3$, and $p = 0.2$. The probability entry is 0.0579. The desired probability is 1 - 0.0579 = 0.9421.

In general, there are k or more successes in n trials iff there are not $n - k + 1$ or more failures.

m-functions for binomial probabilities

Although tables are convenient for calculation, they impose serious limitations on the available parameter values, and when the values are found in a table, they must still be entered into the problem. Fortunately, we have convenient m-functions for these distributions. When MATLAB is available, it is much easier to generate the needed probabilities than to look them up in a table, and the numbers are entered directly into the MATLAB workspace. And we have great freedom in selection of parameter values. For example we may use n of a thousand or more, while tables are usually limited to n of 20, or at most 30. The two m-functions for calculating $P(A_{kn}$ and $P(B_{kn}$ are

$P(A_{kn})$ is calculated by `y = ibinom(n,p,k)`, where k is a row or column vector of integers between 0 and n. The result y is a row vector of the same size as k.

$P(B_{kn})$ is calculated by `y = cbinom(n,p,k)`, where k is a row or column vector of integers between 0 and n. The result y is a row vector of the same size as k.

Example 4.3.7 Use of m-functions ibinom and cbinom

If $n = 10$ and $p = 0.39$, determine $P(A_{kn})$ and $P(B_{kn})$ for $k = 3, 5, 6, 8$.

```
>> p = 0.39;
>> k = [3 5 6 8];
>> Pi = ibinom(10,p,k)  % individual probabilities
Pi = 0.2237    0.1920    0.1023    0.0090
>> Pc = cbinom(10,p,k)  % cumulative probabilities
Pc = 0.8160    0.3420    0.1500    0.0103
```

Note that we have used probability $p = 0.39$. It is quite unlikely that a table will have this probability. Although we use only $n = 10$, frequently it is desirable to use values of several hundred. The m-functions work well for n up to 1000 (and even higher for small values of p or for values very near to one). Hence, there is great freedom from the limitations of tables. If a table with a specific range of values is desired, an m-procedure called *binomial* produces such a table. The use of large n raises the question of cumulation of errors in sums or products. The level of precision in MATLAB calculations is sufficient that such roundoff errors are well below pratical concerns.

Example 4.3.8

```
>> binomial                              % call for procedure
Enter n, the number of trials  13
Enter p, the probability of success  0.413
Enter row vector k of success numbers  0:4
    n            p
   13.0000    0.4130
       k      P(X=k)    P(X>=k)
```

```
         0    0.0010    1.0000
    1.0000    0.0090    0.9990
    2.0000    0.0379    0.9900
    3.0000    0.0979    0.9521
    4.0000    0.1721    0.8542
```

Remark. While the m-procedure binomial is useful for constructing a table, it is usually not as convenient for problems as the m-functions ibinom or cbinom. The latter calculate the desired values and *put them directly into the* MATLAB *workspace*.

Joint Bernoulli trials

Bernoulli trials may be used to model a variety of practical problems. One such is to compare the results of two sequences of Bernoulli trials carried out independently. The following simple example illustrates the use of MATLAB for this.

Example 4.3.9 A joint Bernoulli trial

Bill and Mary take ten basketball free throws each. We assume the two seqences of trials are independent of each other, and each is a Bernoulli sequence.

Mary: Has probability 0.80 of success on each trial.

Bill: Has probability 0.85 of success on each trial.

What is the probability Mary makes more free throws than Bill?

Solution

We have two Bernoulli sequences, operating independently.

Mary: $n = 10$, $p = 0.80$

Bill: $n = 10$, $p = 0.85$

Let

M be the event Mary wins

M_k be the event Mary makes k or more freethrows.

B_j be the event Bill makes exactly j reethrows

Then Mary wins if Bill makes none and Mary makes one or more, or Bill makes one and Mary makes two or more, etc. Thus

$$M = B_0 M_1 \bigvee B_1 M_2 \bigvee \cdots \bigvee B_9 M_{10}$$

and

$$P(M) = P(B_0)P(M_1) + P(B_1)P(M_2) + \cdots + P(B_9)P(M_{10})$$

We use cbinom to calculate the cumulative probabilities for Mary and ibinom to obtain the individual probabilities for Bill.

```
>> pm = cbinom(10,0.8,1:10);     % cumulative probabilities for Mary
>> pb = ibinom(10,0.85,0:9);     % individual probabilities for Bill
>> D = [pm; pb]'                 % display: pm in the first column
   D =                           % pb in the second column
    1.0000    0.0000
    1.0000    0.0000
    0.9999    0.0000
    0.9991    0.0001
    0.9936    0.0012
    0.9672    0.0085
```

```
    0.8791    0.0401
    0.6778    0.1298
    0.3758    0.2759
    0.1074    0.3474
```

To find the probability $P(M)$ that Mary wins, we need to multiply each of these pairs together, then sum. This is just the dot or scalar product, which MATLAB calculates with the command $pm * pb'$. We may combine the generation of the probabilities and the multiplication in one command:

```
>> P = cbinom(10,0.8,1:10)*ibinom(10,0.85,0:9)'
   P = 0.273
```

The ease and simplicity of calculation with MATLAB make it feasible to consider the effect of different values of *n*. Is there an optimum number of throws for Mary? Why should there be an optimum?

An alternate treatment of this problem in the unit on Independent Random Variables utilizes techniques for independent simple random variables.

Alternate MATLAB implementations

Alternate implementations of the functions for probability calculations are found in the *Statistical Package* available as a supplementary package. We have utilized our formulation, so that only the basic MATLAB package is needed.