

4.2 Discrete-Data Histograms

Given a univariate sample $\mathcal{S} = \{x_1, x_2, \dots, x_n\}$, if the sample size is sufficiently large (say $n > 50$ or so) then generally it is meaningful to do more than just calculate the sample mean and standard deviation. In particular, the sample can be processed to form a *histogram* and thereby gain insight into the distribution of the data. There are two cases to consider, discrete data and continuous data. In this section we deal with the easier case, discrete data; continuous data is considered in the next section.

4.2.1 DISCRETE-DATA HISTOGRAMS

Definition 4.2.1 Given the discrete-data sample (multiset) $\mathcal{S} = \{x_1, x_2, \dots, x_n\}$, let \mathcal{X} be the set of possible distinct values in \mathcal{S} . For each $x \in \mathcal{X}$ the *relative frequency (proportion)* is

$$\hat{f}(x) = \frac{\text{the number of } x_i \in \mathcal{S} \text{ for which } x_i = x}{n}$$

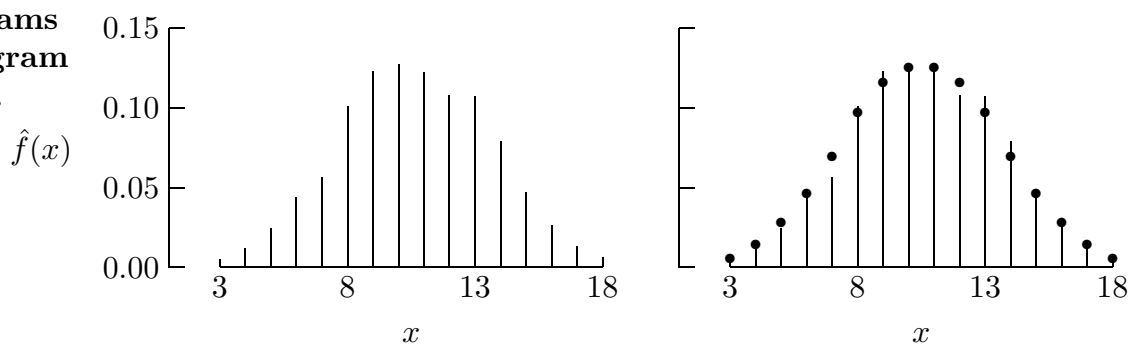
A *discrete-data histogram* is a graphical display of $\hat{f}(x)$ versus x .*

The point here is that if the sample size $n = |\mathcal{S}|$ is large and the data is discrete then it is reasonable to expect that the number of distinct values in the sample will be much smaller than the sample size — values will appear multiple times. A discrete-data histogram is nothing more than a graphical display of the relative frequency with which each distinct value in the sample appears.

Example 4.2.1 As an example of a discrete-data histogram, a modified version of program `galileo` was used to replicate $n = 1000$ rolls of three fair dice (with an `rng` initial seed of 12345) and thereby create a discrete-data sample $\mathcal{S} = \{x_1, x_2, \dots, x_n\}$ with each x_i an integer between 3 and 18 inclusive. (Each x_i is the sum of the three up faces.) Therefore, $\mathcal{X} = \{3, 4, \dots, 18\}$. The resulting relative frequencies (probability estimates) are plotted as the discrete-data histogram on the left side of Figure 4.1.2. Illustrated on the right is the same histogram with the theoretical probabilities associated with the sum-of-three-dice experiment superimposed as \bullet 's. Each theoretical probability represents the limit to which the corresponding relative frequency will converge as $n \rightarrow \infty$.

Figure 4.2.1.

**Histograms
for program
galileo.**



* The reason for the $\hat{f}(x)$ “hat” notation will be explained in Chapter 8.

No matter how many replications are used, in Example 4.2.1 we know *a priori* that $\mathcal{X} = \{3, 4, \dots, 18\}$. Because of that it is natural to use an array to accumulate the relative frequencies, as in program `galileo`. An array can also be used to accumulate the relative frequencies in the next example. In this case, however, the use of an array is less natural because, although we know that $\mathcal{X} = \{0, 1, 2, \dots, b\}$ for some value of b , a reasonable value for b , valid for *any* value of n , is not easily established a priori. If an array is not appropriate, a more flexible data structure based on dynamic memory allocation is required to tally relative frequencies. As an illustration, a linked-list implementation of a discrete-data histogram algorithm is presented later in this section.

Example 4.2.2 Suppose that $2n = 2000$ balls are placed *at random* into $n = 1000$ boxes. That is, for each ball a box is selected at random and the ball is placed in it. The Monte Carlo simulation algorithm given below can be used to generate a random sample $\mathcal{S} = \{x_1, x_2, \dots, x_n\}$ where, for each i , x_i is the number of balls placed in box i .

```

n = 1000;
for (i = 1; i <= n; i++)                               /* i counts boxes */
    xi = 0;
for (j = 0; j < 2 * n; j++) {                          /* j counts balls */
    i = Equilikely(1, n);                               /* pick a box at random */
    xi++;                                             /* then put a ball in it */
}
return x1, x2, ..., xn;

```

When 2000 balls are placed into 1000 boxes in this way then each box will have exactly two balls in it on average. Some boxes will be empty, however, some will have just one ball, some will have two balls, some will have three balls, etc., as illustrated in Figure 4.2.2.

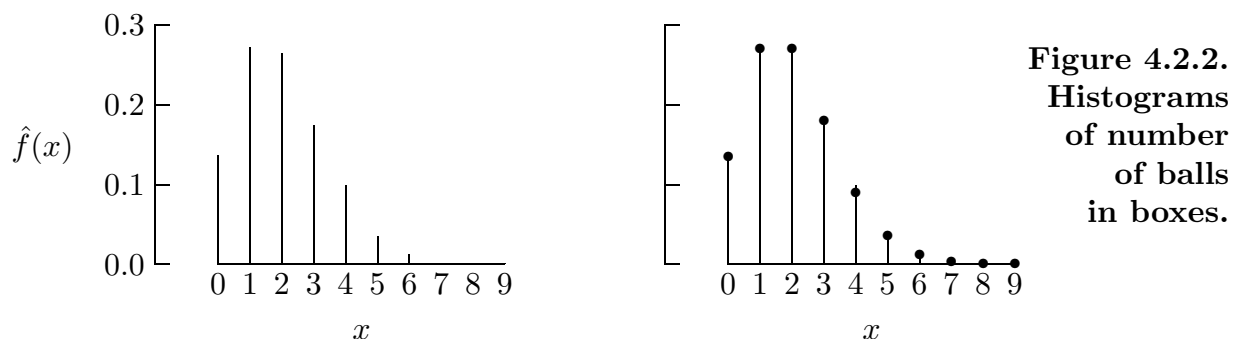


Figure 4.2.2.
Histograms
of number
of balls
in boxes.

In this case, for an `rng` initial seed of 12345, $\mathcal{X} = \{0, 1, 2, \dots, 9\}$. As in Example 4.2.1, the discrete-data histogram is on the left and the figure on the right is the same histogram with theoretical probabilities superimposed as \bullet 's. Each theoretical probability represents the limit to which the corresponding relative frequency will converge as $n \rightarrow \infty$. (The relative frequencies $\hat{f}(7) = 0.002$ and $\hat{f}(9) = 0.001$ are too small to be visible in the histogram and, because no boxes ended up with 8 balls, $\hat{f}(8) = 0.000$.)

Note the asymmetric shape of the histogram in Example 4.2.2. This asymmetry is frequently associated with non-negative data when the mean is small. As an exercise you should repeat this example in the situation where there are 1000 boxes and 10 000 balls. In that case the shape of the histogram will change to a distribution symmetric about the expected number of balls in each box, which is 10. The reason for this shift to a symmetric distribution will be discussed in Chapter 6.

Histogram Mean and Standard Deviation

Definition 4.2.2 Given the relative frequencies from Definition 4.2.1, the *discrete-data histogram mean* is

$$\bar{x} = \sum_x x \hat{f}(x)$$

and the associated *discrete-data histogram standard deviation* is

$$s = \sqrt{\sum_x (x - \bar{x})^2 \hat{f}(x)}$$

where the sum is over all $x \in \mathcal{X}$. The discrete-data histogram variance is s^2 .

Consistent with their interpretation as probability estimates, the relative frequencies from Definition 4.2.1 are defined so that $\hat{f}(x) \geq 0$ for all $x \in \mathcal{X}$ and

$$\sum_x \hat{f}(x) = 1.$$

Moreover, it follows from the definition of \mathcal{S} and \mathcal{X} that

$$\sum_{i=1}^n x_i = \sum_x x n \hat{f}(x) \quad \text{and} \quad \sum_{i=1}^n (x_i - \bar{x})^2 = \sum_x (x - \bar{x})^2 n \hat{f}(x).$$

(In both equations, the two summations compute the same thing, but in a different order.) Therefore, from the equations in Definition 4.1.1 and Definition 4.2.2 it follows that the *sample* mean and standard deviation are mathematically equivalent to the *discrete-data histogram* mean and standard deviation, respectively.

Provided the relative frequencies have already been computed, it is generally preferable to compute \bar{x} and s using the discrete-data histogram equations. Moreover, the histogram-based equations have great theoretical significance in that they provide the motivation for defining the mean and standard deviation of discrete random variables — see Chapter 6.

The equation for s in Definition 4.2.2 is the two-pass version of the standard deviation equation. The mathematically equivalent one-pass version of this equation is

$$s = \sqrt{\left(\sum_x x^2 \hat{f}(x) \right) - \bar{x}^2}$$

where the sum is over all $x \in \mathcal{X}$.

Example 4.2.3 For the data in Example 4.2.1

$$\bar{x} = \sum_{x=3}^{18} x \hat{f}(x) \cong 10.609 \quad \text{and} \quad s = \sqrt{\sum_{x=3}^{18} (x - \bar{x})^2 \hat{f}(x)} \cong 2.925$$

Similarly, for the data in Example 4.2.2

$$\bar{x} = \sum_{x=0}^9 x \hat{f}(x) = 2.0 \quad \text{and} \quad s = \sqrt{\sum_{x=0}^9 (x - \bar{x})^2 \hat{f}(x)} \cong 1.419$$

4.2.2 COMPUTATIONAL CONSIDERATIONS

As illustrated in Examples 4.2.1 and 4.2.2, in many simulation applications the discrete data will, in fact, be *integer*-valued. For integer-valued data, the usual way to tally the discrete-data histogram is to use an array. Recognize, however, that the use of an array data structure for the histogram involves the allocation of memory with an associated requirement to know the range of data values. That is, when memory is allocated it is necessary to assume that $\mathcal{X} = \{a, a + 1, a + 2, \dots, b\}$ for reasonable integer values of $a < b$. Given a knowledge of a, b the following one-pass $O(n)$ algorithm is the preferred way to compute a discrete-data histogram for integer-valued data.

Algorithm 4.2.1 Given integers a, b with $a < b$ and integer-valued data x_1, x_2, \dots the following algorithm computes a discrete-data histogram.

```

long count[b - a + 1];
n = 0;
for (x = a; x <= b; x++)
    count[x - a] = 0;
outliers.lo = 0;
outliers.hi = 0;
while ( more data ) {
    x = GetData();
    n++;
    if ((a <= x) and (x <= b))
        count[x - a]++;
    else if (a > x)
        outliers.lo++;
    else
        outliers.hi++;
}
return n, count[], outliers           /*  $\hat{f}(x)$  is (count[x - a] / n) */

```

Outliers

By necessity Algorithm 4.2.1 allows for the possibility of *outliers* — occasional x_i that fall outside the range $a \leq x_i \leq b$. Generally with simulation-generated data there should not be any outliers. That is, an outlier is a data point that some omniscient being considers to be so different from the rest that it should be excluded from any statistical analysis. Although outliers are common with some kinds of experimentally measured data, it is difficult to argue that *any* data generated by a *valid* discrete-event simulation program is an outlier, no matter how unlikely the value may appear to be.

General-Purpose Discrete-Data Histogram Algorithm

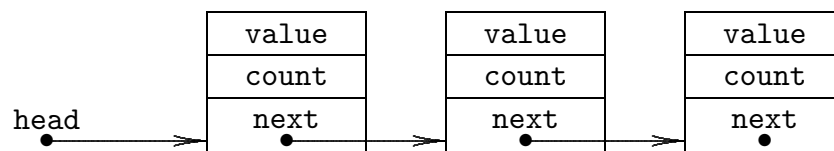
Algorithm 4.2.1 is not appropriate as a *general-purpose* discrete-data histogram algorithm for two reasons.

- If the discrete-data sample is integer-valued and the a , b parameters are not chosen properly then either outliers will be produced, perhaps without justification, or excessive computational memory (to store the `count` array) may be required needlessly.
- If the data is not integer-valued then Algorithm 4.2.1 is not applicable.

In either case we must find an alternative, more general, algorithm that uses dynamic memory allocation and has the ability to handle both real-valued and integer-valued data. The construction of this algorithm is one of several occasions in this book where we will use dynamic memory allocation, in this case in the form of a linked-list, as the basis for an algorithm that is well suited to accommodate the uncertainty and variability naturally associated with discrete-event simulation data.

Definition 4.2.3 A linked-list discrete-data histogram algorithm with general applicability can be constructed using a *list pointer* `head` and multiple *list nodes* (structures) each consisting of three fields `value`, `count`, and `next`, as illustrated in Figure 4.2.3.

Figure 4.2.3.
Linked list data structure for discrete data.



The association between the first two fields in each list node and the corresponding terms in Definition 4.2.1 is $x \sim \text{value}$ and, after all data is processed, $n\hat{f}(x) \sim \text{count}$. The `next` field is the link (pointer) from one list node to the next.

Algorithm 4.2.2 Given the linked-list data structure in Definition 4.2.3 and the discrete data x_1, x_2, \dots, x_n a discrete-data histogram is computed by using x_1 to initialize the first list node. Then, for $i = 2, 3, \dots, n$, as each x_i is read the list is searched (linearly from the head by following the links) to see if a list node with `value` equal to x_i is already present in the list. If so the corresponding list node `count` is increased by 1; otherwise a new list node is added to the end of the list with `value` equal to x_i and `count` equal to 1.

Example 4.2.4 For the discrete data 3.2, 3.7, 3.7, 2.9, 3.7, 3.2, 3.7, 3.2, Algorithm 4.2.2 generates the corresponding linked-list illustrated in Figure 4.2.4.

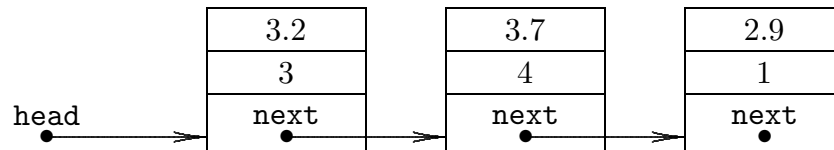


Figure 4.2.4.
Sample linked list.

Note that the order of the linked-list nodes is dictated by the order in which the data appears in the sample. As an alternative, it may be better to use the `count` field and maintain the list in *sorted* order by decreasing relative frequency, or to use the `value` field and sort by data value — see Exercise 4.2.6.

Program ddh

The discrete-data histogram program `ddh` is based on Algorithm 4.2.2 and the linked-list data structure in Definition 4.2.3. The program has been designed so that the linked list is sorted by value prior to output. This program is valid for both integer-valued and real-valued input with no artificial outlier check imposed and no restriction on the sample size. Like program `uvs`, program `ddh` supports file redirection, as illustrated by Example 4.2.5 (to follow).

If program `ddh` is used improperly to tally a large sample that is *not* discrete the program's execution time may be excessive. That is, the design of program `ddh` is based on the assumption that, even though the sample size $|\mathcal{S}|$ is essentially arbitrary, the number of distinct values in the sample $|\mathcal{X}|$ is not too large, say a few hundred or less. If the number of distinct values is large then the $O(|\mathcal{X}|)$ complexity (per sample value) of the function `Insert` and the one-time $O(|\mathcal{X}|^2)$ complexity of the function `Sort` will become the source of excessive execution time. Note, however, that a discrete-data sample with more than a few hundred distinct values would be *very* unusual.

Example 4.2.5 Program `sis2` computes the statistics \bar{d} , \bar{o} , \bar{u} , \bar{l}^+ , and \bar{l}^- . If we were interested in a more detailed look at this simple inventory system we could, for example, construct a histogram of the inventory level prior to inventory review. This is easily accomplished by eliminating the printing of summary statistics in program `sis2` and modifying the `while` loop in `main` by inserting the one line indicated

```

index++;
printf("%ld\n", inventory);          /* this line is new */
if (inventory < MINIMUM) {

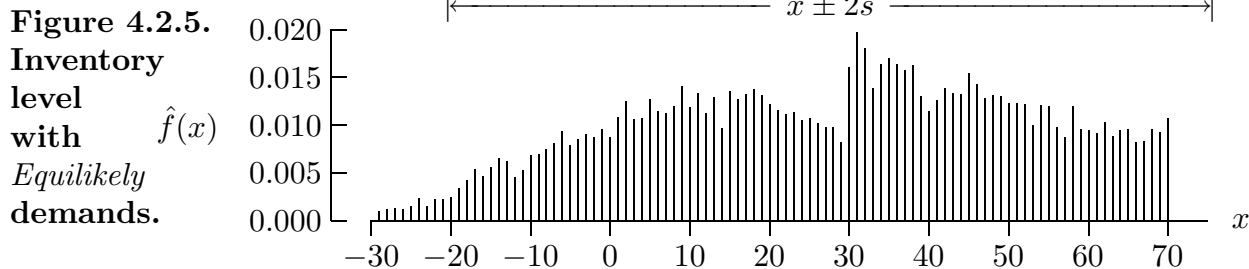
```

If the new program `sis2` is compiled to disk (with `STOP = 10 000`), the command

```
sis2 | ddh > sis2.out
```

will then produce a discrete-data histogram file `sis2.out` from which an inventory level histogram can be constructed.

Example 4.2.6 As a continuation of Example 4.2.5 the inventory level histogram is illustrated (x denotes the inventory level prior to review) in Figure 4.2.5.



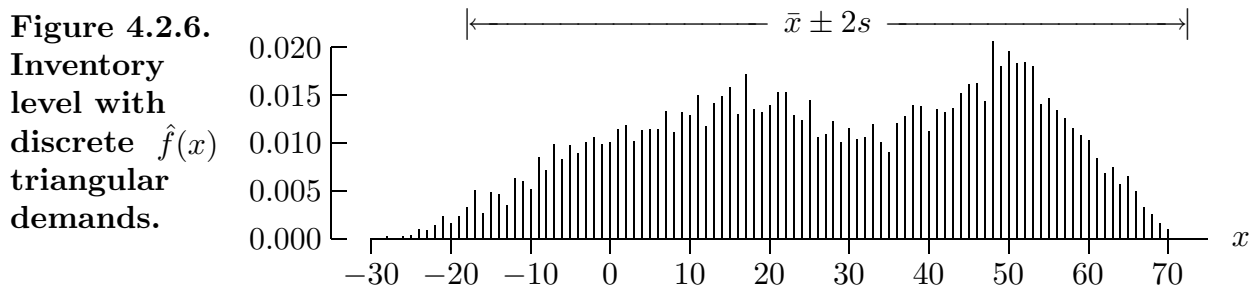
The sample mean and standard deviation for this data are $\bar{x} = 27.63$ and $s = 23.98$. The corresponding $\bar{x} \pm 2s$ interval is indicated. In this case approximately 98.5% of the data falls in this interval. Therefore, consistent with the discussion in the previous section, the “at least 75%” estimate guaranteed by Chebyshev’s inequality is very conservative. That is, the $\bar{x} \pm 2s$ interval contains almost all the sample data.

As discussed in Example 3.1.5, the assumption of *Equilikely*(10, 50) random variate demands in program `sis2` is questionable. Generally one would expect the extreme values of the demand, in this case 10 and 50, to be much less likely than an intermediate value like 30. One way to accomplish this is use the assignment

```
return(Equilikely(5, 25) + Equilikely(5, 25));
```

in the function `GetDemand`. It can be verified by the axiomatic approach to probability that the theoretical mean and standard deviation of the demand for this assignment is 30 and $\sqrt{220/3} \cong 8.56$ respectively. Moreover, the distribution and associated histogram will have a triangular shape with a peak at 30. Of course, other stochastic demand models are possible as well. We will investigate this issue in more detail in Chapter 6.

Example 4.2.7 Compared to the histogram in Example 4.2.6, the use of the more realistic discrete triangular random variate demand model produces a corresponding change in the inventory level histogram, as illustrated in Figure 4.2.6.

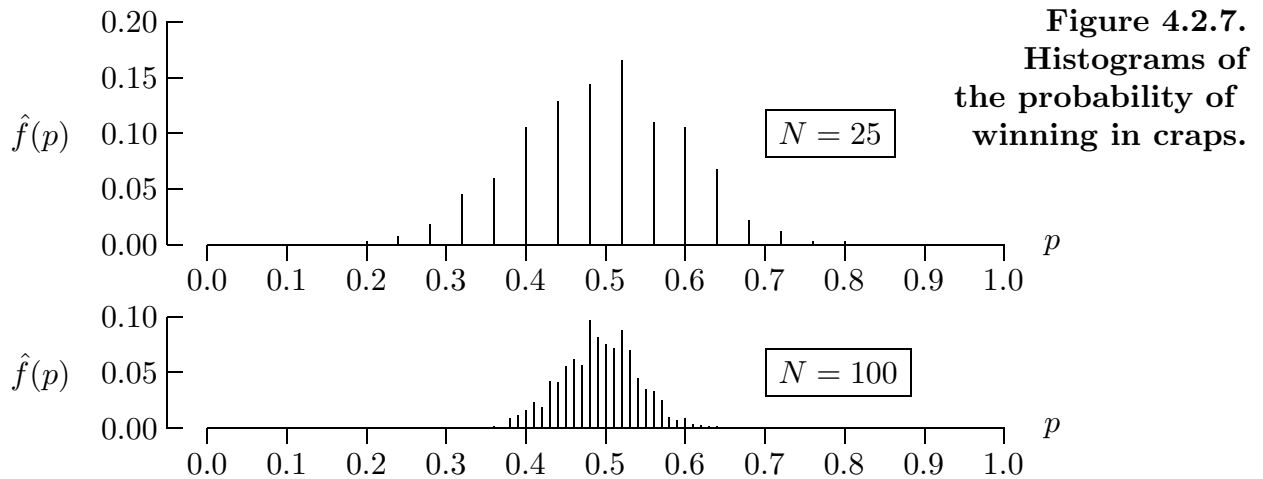


This inventory level histogram is much more tapered at the extreme values, particularly at 70. The sample mean and standard deviation are 27.29 and 22.59 respectively. The corresponding $\bar{x} \pm 2s$ interval is slightly smaller, as illustrated. As in Example 4.2.6, approximately 98.5% of the data lie in this interval.

Accuracy of Point Estimates

We now consider the accuracy of probability estimates derived by Monte Carlo simulation. This issue was first raised in Chapter 2 and, at that point, largely dismissed. Now we can use program `ddh` as an experimental tool to study the inherent uncertainty in such probability estimates. The following example uses the Monte Carlo program `craps` from Section 2.4 to generate 1000 point estimates of the probability of winning in the dice game craps. Because of the inherent uncertainty in any one of these estimates, when many estimates are generated a natural distribution of values will be produced. This distribution of values can be characterized with a discrete-data histogram and in that way we can gain significant insight into just how uncertain any one probability estimate may be.

Example 4.2.8 A Monte Carlo simulation of the dice game craps was used to generate 1000 estimates of the probability of winning. In the top figure each estimate is based on just $N = 25$ plays of the game; in the bottom figure $N = 100$ plays per estimate were used. For either figure, let $p_1, p_2, p_3, \dots, p_n$ denote the $n = 1000$ estimates where $p_i = x_i/N$ and x_i is the number of wins in N plays of the game. Since only $N + 1$ values of p_i are possible, it is natural to use a discrete-data histogram in this application, as illustrated in Figure 4.2.7.



In the top histogram, the sample mean is 0.494, the sample standard deviation is 0.102 and there is significant variation about the true probability of winning at craps, which is known to be $244/495 \cong 0.4929$ using the analytic approach to probability. This variation corresponds to an inherently large uncertainty in any one estimate. That is, if just *one* probability estimate were generated based on 25 plays of the game, then our experiment with 1000 replications shows that probability estimates as far away from the true probability as $5/25 = 0.2$ and $20/25 = 0.8$ could be generated. In the bottom histogram, the associated sample mean and standard deviation are 0.492 and 0.048 respectively. When compared with the top histogram there is still significant variability, but it is reduced. In particular, the standard deviation is reduced by a factor of about two (from 0.102 to 0.048) resulting in a corresponding two-fold reduction in the uncertainty of any one estimate.

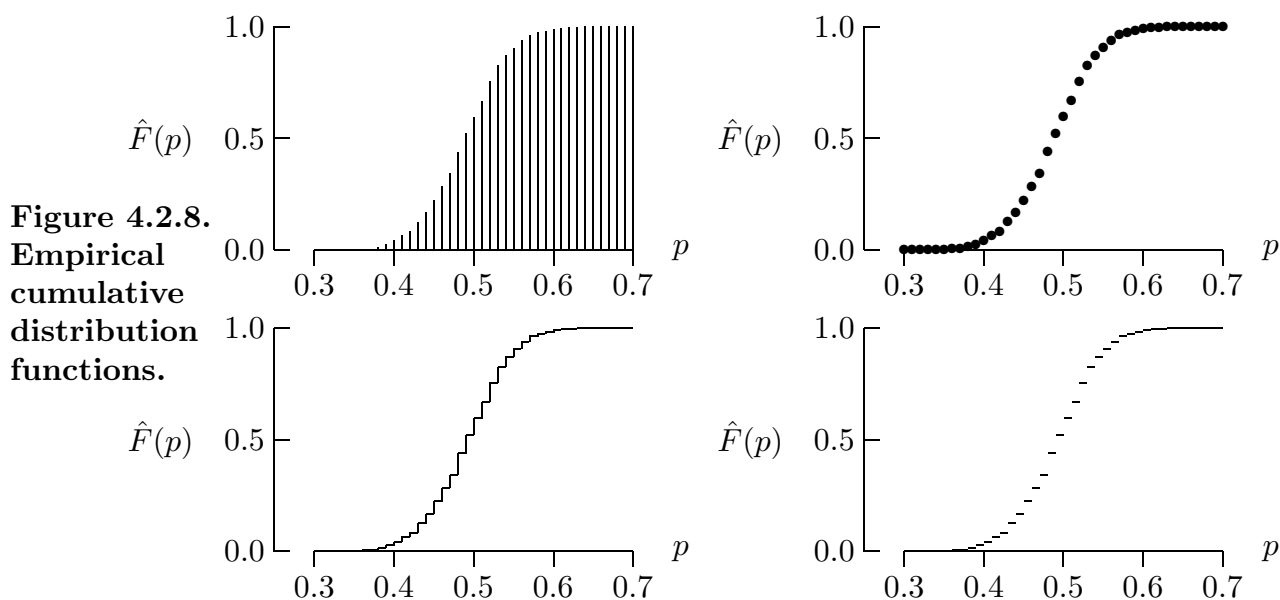
The reduction in the sample standard deviation from 0.102 to 0.048 in Example 4.2.8 is good. That a *four-fold* increase in the number of replications (games) is required to produce a *two-fold* reduction in uncertainty, however, is not so good. We will have much more to say about this in Chapter 8.

In Example 4.2.8 the “bell-shape” of the discrete-data histogram is important. As will be discussed in Chapter 8, this bell (or Gaussian) shape, which shows up in a wide variety of applications, is accurately described by a well-known mathematical equation that provides the basis for establishing a probability estimate *interval* within which the true (theoretical) probability lies with high confidence.

4.2.3 EMPIRICAL CUMULATIVE DISTRIBUTION FUNCTIONS

The histogram is an effective tool for estimating the shape of a distribution. There are occasions, however, when a cumulative version of the histogram is preferred. Two such occasions are when quantiles (e.g., the 90th quantile of a distribution) are of interest or when two or more distributions are to be compared. An *empirical cumulative distribution function*, which simply takes an upward step of $1/n$ at each of the n data values x_1, x_2, \dots, x_n , is easily computed. In the case of discrete data, of course, there will typically be many ties, so the step function will have varying heights to the risers.

Example 4.2.9 The $n = 1000$ estimates of the probability of winning in craps for $N = 100$ plays from Example 4.2.8 range from 0.33 to 0.64. Four *empirical cumulative distribution functions* for this data set using different formats are plotted in Figure 4.2.8 for $0.3 < p < 0.7$. The top two graphs plot the cumulative probability only at the observed values, while the bottom two graphs are step functions plotted for all values of p . The choice between these four styles is largely a matter of taste.



4.2.4 EXERCISES

Exercise 4.2.1^a Find the theoretical values of the probabilities, mean, and variance for program `galileo` in Example 4.2.1.

Exercise 4.2.2 (a) Generate the 2000-ball histogram in Example 4.2.2. (b) Verify that the resulting relative frequencies $\hat{f}(x)$ satisfy the equation

$$\hat{f}(x) \cong \frac{2^x \exp(-2)}{x!} \quad x = 0, 1, 2, \dots$$

(c) Then generate the corresponding histogram if 10 000 balls are placed, at random, in 1000 boxes. (d) Find an equation that seems to fit the resulting relative frequencies well and illustrate the quality of the fit.

Exercise 4.2.3 The gap in the histogram in Figure 4.2.2 in Example 4.2.2 corresponding to no boxes having exactly eight balls brings up an important topic in Monte Carlo and discrete-event simulation: the generation of *rare events*. To use the probabilist's terminology, there is poorer precision for estimating the probability of events out in the fringes (or "tails") of a distribution. (a) In theory, could all of the 2000 balls be in just one of the boxes? If so, give the probability of this event. If not, explain why not. (b) In practice, with our random number generator `rng`, could all of the 2000 balls be in one of the boxes?

Exercise 4.2.4^a Find the theoretical probabilities required to generate the right-hand plot in Figure 4.2.2.

Exercise 4.2.5 Program `ddh` computes the *histogram* mean and standard deviation. If you were to modify this program so that it also computes the *sample* mean and standard deviation using Welford's algorithm you would observe that (except perhaps for floating-point round-off errors) these two ways of computing the mean and standard deviation produce identical results. (a) If, instead, Algorithm 4.2.1 were used to compute the histogram mean and standard deviation, would they necessarily agree exactly with the sample mean and standard deviation? (b) Why or why not?

Exercise 4.2.6 Although the output of program `ddh` is sorted by `value`, this is only a convention. As an alternative, modify program `ddh` so that the output is sorted by `count` (in decreasing order).

Exercise 4.2.7^a Use simulation to explore the "efficiency increasing" modifications suggested in Example 4.2.4. That is, either of these modifications will make the algorithm more complicated for what may be, in return, only a marginal increase in efficiency. Because of its quadratic complexity beware of simply using a modified version of the function `Sort` in this application.

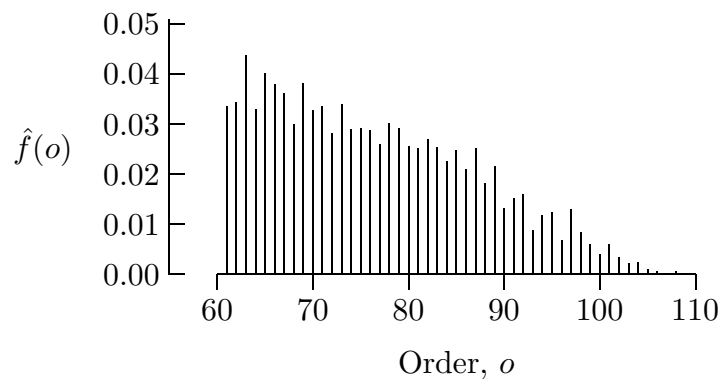
Exercise 4.2.8 How do the inventory level histograms in Examples 4.2.6 and 4.2.7 relate to the relative frequency of setups \bar{u} ?

Exercise 4.2.9^a Generate a random variate demand sample of size $n = 10\,000$ as

$$d_i = \text{Equilikely}(5, 25) + \text{Equilikely}(5, 25); \quad i = 1, 2, \dots, n$$

- (a) Why is the sample mean 30? (b) Why is the the sample standard deviation $\sqrt{220/3}$?
 (c) Why is the shape of the histogram triangular?

Exercise 4.2.10 A discrete-data histogram of orders is illustrated (corresponding to the demand distribution in Example 4.2.6). Based on this histogram, *by inspection* estimate the histogram mean and standard deviation.



Exercise 4.2.11^a A test is compiled by selecting 12 different questions, at random and without replacement, from a well-publicized list of 120 questions. After studying this list you are able to classify all 120 questions into two classes, I and II. Class I questions are those about which you feel confident; the remaining questions define class II. Assume that your grade probability, conditioned on the class of the problems, is

	<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>F</i>
class I	0.6	0.3	0.1	0.0	0.0
class II	0.0	0.1	0.4	0.4	0.1

Each test question is graded on an $A = 4$, $B = 3$, $C = 2$, $D = 1$, $F = 0$ scale and a score of 36 or better is required to pass the test. (a) If there are 90 class I questions in the list, use Monte Carlo simulation and 100 000 replications to generate a discrete-data histogram of scores. (b) Based on this histogram what is the probability that you will pass the test?

Exercise 4.2.12^a Modify program `ssq2` so that each time a new job arrives the program outputs the number of jobs in the service node *prior* to the job's arrival. (a) Generate a discrete-data histogram for 10 000 jobs. (b) Comment on the shape of the histogram and compare the histogram mean with \bar{l} . (c) How does $\hat{f}(0)$ relate to \bar{x} ?