Section 4.2: Discrete-Data Histograms
Given a discrete-data sample multiset $S = \{x_1, x_2, \ldots, x_n\}$ with possible values $\mathcal{X}$, the relative frequency is

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

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

If $n = |S|$ is large relative to $|\mathcal{X}|$ then values will appear multiple times.
Example 4.2.1

Program galileo was used to replicate \( n = 1000 \) rolls of three dice

- Discrete-data sample is \( S = \{x_1, x_2, \ldots, x_{1000}\} \)
- Each \( x_i \) is an integer between 3 and 18, so \( \mathcal{X} = \{3, 4, \ldots, 18\} \)
- Theoretical probabilities: \( \hat{f}'s \) (limit as \( n \to \infty \))

Since \( \mathcal{X} \) is known \textit{a priori}, use an array
Suppose $2n = 2000$ balls are placed \textit{at random} into $n = 1000$ boxes:

```c
n = 1000;
for (i = 1; i <= n; i++) /* i counts boxes */
    x_i = 0;
for (j = 1; i <= 2*n; i++) {
    i = Equilike(1, n); /* pick a box at random */
    x_i++;
}
return x_1, x_2, ..., x_n;
```
Example 4.2.2

- $S = \{x_1, x_2, \ldots, x_n\}$, $x_i$ is the number of balls placed in box $i$
- $\bar{x} = 2.0$
- Some boxes will be empty, some will have one ball, some will have two balls, etc.
- For seed 12345:
The discrete-data histogram mean is

\[ \bar{x} = \sum_x x \hat{f}(x) \]

The discrete-data histogram standard deviation is

\[ s = \sqrt{\sum_x (x - \bar{x})^2 \hat{f}(x)} \text{ or } s = \sqrt{\left( \sum_x x^2 \hat{f}(x) \right) - \bar{x}^2} \]

The discrete-data histogram variance is \( s^2 \)
By definition, $\hat{f}(x) \geq 0$ for all $x \in \mathcal{X}$ and

$$\sum_{x} \hat{f}(x) = 1$$

From the definition of $S$ and $\mathcal{X}$,

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

The sample mean/standard deviation is mathematically equivalent to the discrete-data histogram mean/standard deviation

If frequencies $\hat{f}(\cdot)$ have already been computed, $\bar{x}$ and $s$ should be computed using discrete-data histogram equations
Example 4.2.3

- For the data in Example 4.2.1 (three dice):

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

- For the data in Example 4.2.2 (balls placed in boxes):

  \[
  \bar{x} = \sum_{x=0}^{9} x \hat{f}(x) = 2.0 \quad \text{and} \quad s = \sqrt{\sum_{x=0}^{9} (x - \bar{x})^2 \hat{f}(x)} \approx 1.419
  \]
Algorithm 4.2.1

Given integers $a, b$ and integer-valued data $x_1, x_2, \ldots$ the following computes a discrete-data histogram:

```plaintext
Algorithm 4.2.1

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) */
```
Algorithm 4.2.1 allows for outliers
- Occasional $x_i$ outside the range $a \leq x_i \leq b$
- Necessary due to array structure

Outliers are common with some experimentally measured data
Generally, *valid* discrete-event simulations should not produce *any* outlier data
Algorithm 4.2.1 is not a good general purpose algorithm:

- For integer-valued data, \( a \) and \( b \) must be chosen properly, or else
  - Outliers may be produced without justification
  - The count array may needlessly require excessive memory
- For data that is not integer-valued, algorithm 4.2.1 is not applicable

We will use a linked-list histogram
Algorithm 4.2.2

- Initialize the first list node, where value = $x_1$ and count = 1
- For all sample data $x_i$, $i = 2, 3, \ldots, n$
  - Traverse the list to find a node with value == $x_i$
  - If found, increase corresponding count by one
  - Else add a new node, with value = $x_i$ and count = 1
Example 4.2.4

- Discrete data sample $S = \{3.2, 3.7, 3.7, 2.9, 3.7, 3.2, 3.7, 3.2\}$
- Algorithm 4.2.2 generates the linked list:

  ![Linked List Diagram]

  Node order is determined by data order in the sample

  Alternatives:
  - Use `count` to maintain the list in order of decreasing frequency
  - Use `value` to sort the list by data value
Program ddh

- Generates a discrete-data histogram, based on algorithm 4.2.2 and the linked-list data structure
- Valid for integer-valued and real-valued input
- No outlier checks
- No restriction on sample size
- Supports file redirection
- Assumes $|\mathcal{X}|$ is small, a few hundred or less
- Requires $\mathcal{O}(|\mathcal{X}|)$ computation per sample value
- Should not be used on continuous samples (see Section 4.3)
Example 4.2.5

Construct a histogram of the inventory level (prior to review) for the simple inventory system

- Remove summary statistics from sis2
- Print the inventory within the while loop in main:

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

- If the new executable is sis2mod, the command
  ```
sis2mod | ddh > sis2.out
  ```
  will produce a discrete-data histogram file sis2.out
Using \texttt{sis2mod} to generate 10,000 weeks of sample data, the inventory level histogram from \texttt{sis2.out} can be constructed.

- $x$ denotes the inventory level prior to review.
- $\bar{x} = 27.63$ and $s = 23.98$
- About 98.5\% of the data falls within $\bar{x} \pm 2s$
Example 4.2.7

Change the demand to \textit{Equilike}{\textit{ly}}(5, 25)+\textit{Equilikely}(5, 25) in \texttt{sis2mod} and construct the new inventory level histogram

- More tapered at the extreme values
- $\bar{x} = 27.29$ and $s = 22.59$
- About 98.5\% of the data falls within $\bar{x} \pm 2s$
Monte Carlo simulation was used to generate 1000 point estimates of the probability of winning the dice game craps.

Sample is \( S = \{p_1, p_2, \ldots, p_{1000}\} \) with
\[
p_i = \frac{\# \text{ wins in } N \text{ plays}}{N}
\]

\( X = \{0/N, 1/N, \ldots, N/N\} \)

Compare \( N = 25 \) and \( N = 100 \) plays per estimate.

\( X \) is small, can use discrete-data histograms.
Histories of Probability of Winning Craps

\[ f(p) \]

- **\( N = 25 \):** \( \bar{p} = 0.494 \), \( s = 0.102 \)
- **\( N = 100 \):** \( \bar{p} = 0.492 \), \( s = 0.048 \)

*Four-fold* increase in number of replications produces *two-fold* reduction in uncertainty.
Empirical Cumulative Distribution Functions

- Histograms estimate distributions
- Sometimes, the *cumulative* version is preferred:
  \[ \hat{F}(x) = \frac{\text{the number of } x_i \in S \text{ with } x_i \leq x}{n} \]
- Useful for quantiles
Example 4.2.9

The empirical cumulative distribution function for $N = 100$ from Example 4.2.8 (in four different styles):