CHAPTER 7

Statistics, Probability, and Interpolation

OUTLINE

This chapter begins with an introduction to basic statistics in Section 7.1. You will see how to obtain and interpret *histograms,* which are specialized plots for displaying statistical results. The *normal distribution,* commonly called the *bell-shaped curve,* forms the basis of much of probability theory and many statistical methods. It is covered in Section 7.2. In Section 7.3 you will see how to include random processes in your simulation programs. In Section 7.4 you will see how to use interpolation with data tables to estimate values that are not in the table.

When you have finished this chapter, you should be able to use MATLAB to do the following:

- Solve basic problems in statistics and probability.
- Create simulations incorporating random processes.
- Apply interpolation techniques.

7.1 Statistics and Histograms

MEAN

MODE

MEDIAN

With MATLAB you can compute the *mean* (the average), the *mode* (the most frequently occurring value), and the *median* (the middle value) of a set of data. MATLAB provides the `mean(x)`, `mode(x)`, and `median(x)` functions to compute the mean, mode, and median of the data values stored in `x`, if `x` is a vector. However, if `x` is a matrix, a row vector is returned containing the mean (or mode or median) value of each column of `x`. These functions do not require the elements in `x` to be sorted in ascending or descending order.

The way the data are spread around the mean can be described by a *histogram* plot. A *histogram* is a plot of the frequency of occurrence of data values versus the values themselves. It is a bar plot of the number of data values that occur within each range, with the bar centered in the middle of the range.

BINS

To plot a histogram, you must group the data into subranges, called *bins*. The choice of the bin width and bin center can drastically change the shape of the histogram. If the number of data values is relatively small, the bin width cannot be small because some of the bins will contain no data and the resulting histogram might not usefully illustrate the distribution of the data.

To obtain a histogram, first sort the data values if they have has not yet been sorted (you can use the `sort` function here). Then choose the bin ranges and bin centers and count the number of values in each bin. Use the `bar` function to plot the number of values in each bin versus the bin centers as a bar chart. The function `bar(x,y)` creates a bar chart of `y` versus `x`.

MATLAB also provides the `hist` command to generate a histogram. This command has several forms. Its basic form is `hist(y)`, where `y` is a vector containing the data. This form aggregates the data into 10 bins evenly spaced between the minimum and maximum values in `y`. The second form is `hist(y,n)`, where `n` is a user-specified scalar indicating the number of bins. The third form is `hist(y,x)`, where `x` is a user-specified vector that determines the location of the bin centers; the bin widths are the distances between the centers.

EXAMPLE 7.1–1 Breaking Strength of Thread

To ensure proper quality control, a thread manufacturer selects samples and tests them for breaking strength. Suppose that 20 thread samples are pulled until they break, and the breaking force is measured in newtons rounded off to integer values. The breaking force values recorded were 92, 94, 93, 96, 93, 94, 95, 96, 91, 93, 95, 95, 95, 92, 93, 94, 91, 94, 92, and 93. Plot the histogram of the data.

■ Solution

Store the data in the vector `y`, which is shown in the following script file. Because there are six outcomes (91, 92, 93, 94, 95, 96 N), we choose six bins. However, if you use `hist(y,6)`, the bins will not be centered at 91, 92, 93, 94, 95, and 96. So use the form

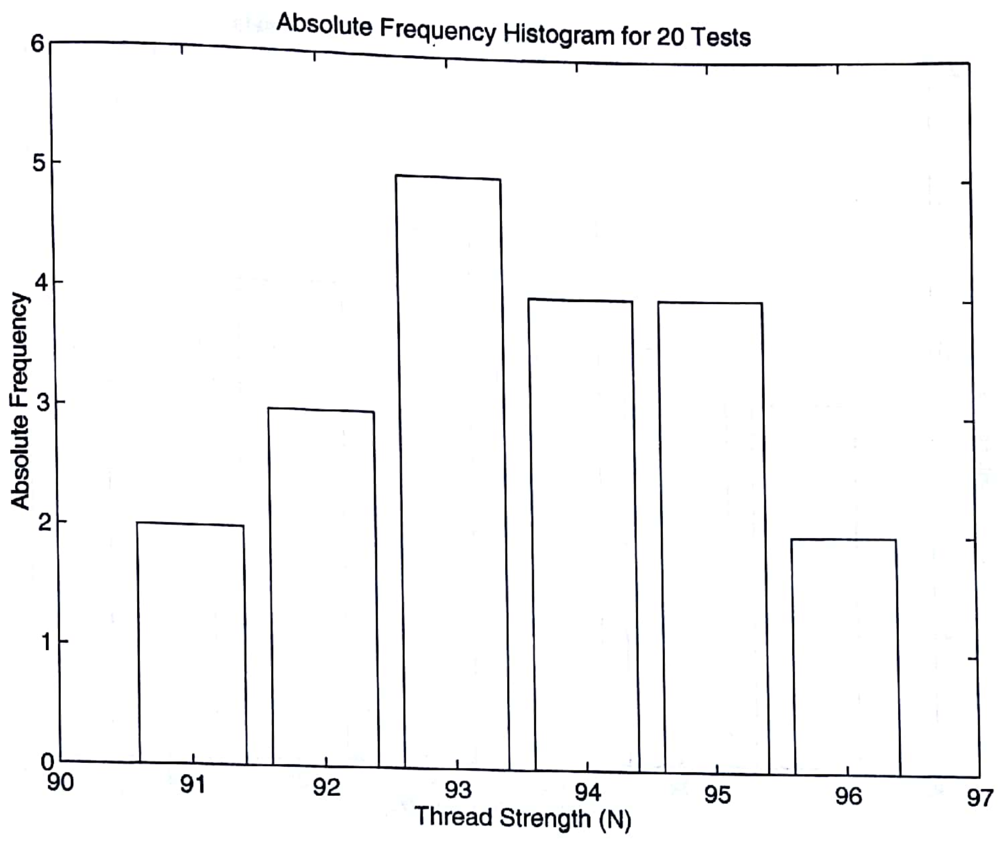


Figure 7.1–1 Histograms for 20 tests of thread strength.

hist(y,x), where x = 91:96. The following script file generates the histogram shown in Figure 7.1–1.

```
% Thread breaking strength data for 20 tests.
y = [92,94,93,96,93,94,95,96,91,93,...
   95,95,95,92,93,94,91,94,92,93];
% The six possible outcomes are 91,92,93,94,95,96.
x = 91:96;
hist(y,x),axis([90 97 0 6]),ylabel('Absolute Frequency'),...
   xlabel('Thread Strength (N)'),...
   title('Absolute Frequency Histogram for 20 Tests')
```

ABSOLUTE FREQUENCY

RELATIVE FREQUENCY

The *absolute frequency* is the number of times a particular outcome occurs. For example, in 20 tests these data show that a 95 occurred 4 times. The absolute frequency is 4, and its *relative frequency* is 4/20, or 20 percent of the time.

When there is a large amount of data, you can avoid typing in every data value by first aggregating the data. The following example shows how this is done using the ones function. The following data were generated by testing 100 thread

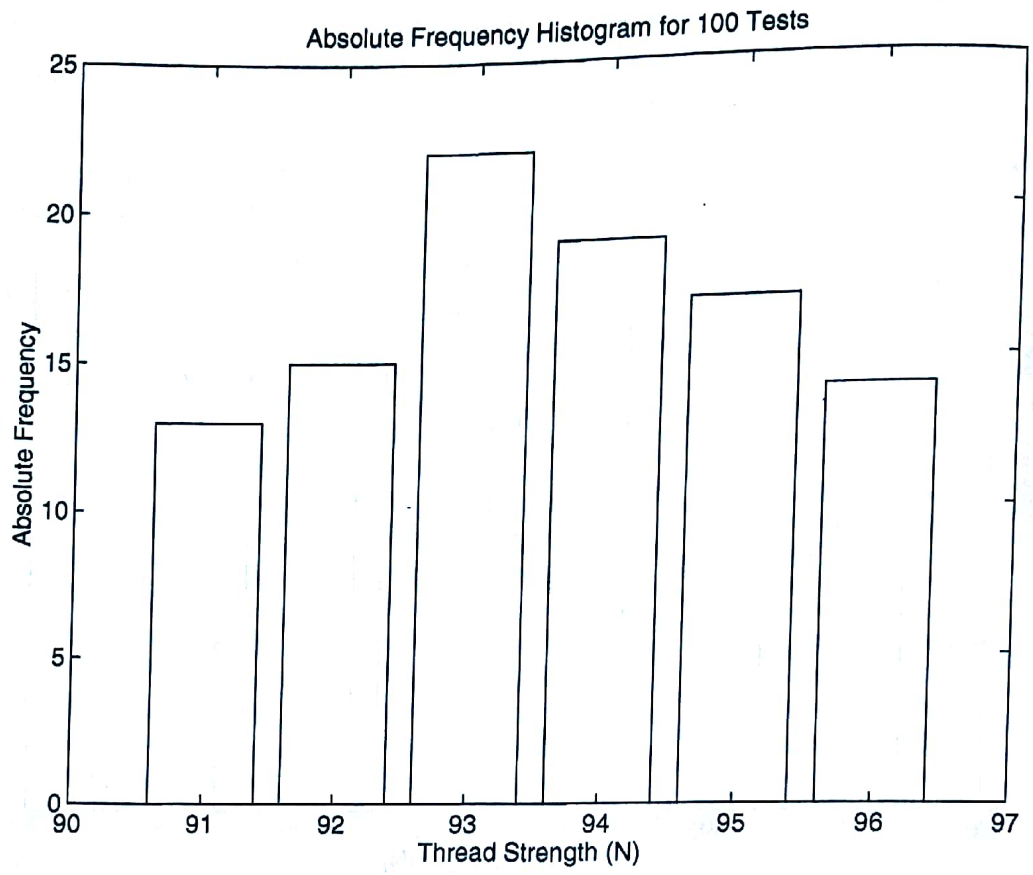


Figure 7.1–2 Absolute frequency histogram for 100 thread tests.

samples. The number of times 91, 92, 93, 94, 95, or 96 N was measured is 13, 15, 22, 19, 17, and 14, respectively.

```
% Thread strength data for 100 tests.
y = [91*ones(1,13),92*ones(1,15),93*ones(1,22),...
   94*ones(1,19),95*ones(1,17),96*ones(1,14)];
x = 91:96;
hist(y,x),ylabel('Absolute Frequency'),...
   xlabel('Thread Strength (N)'),...
   title('Absolute Frequency Histogram for 100 Tests')
```

The result appears in Figure 7.1–2.

The `hist` function is somewhat limited in its ability to produce useful histograms. Unless all the outcome values are the same as the bin centers (as is the case with the thread examples), the graph produced by the `hist` function will not be satisfactory. This case occurs when you want to obtain a *relative* frequency histogram. In such cases you can use the `bar` function to generate the histogram. The following script file generates the relative frequency histogram for the 100 thread tests. Note that if you use the `bar` function, you must aggregate the data first.

```
% Relative frequency histogram using the bar function.
tests = 100;
y = [13,15,22,19,17,14]/tests;
x = 91:96;
bar(x,y),ylabel('Relative Frequency'),...
   xlabel('Thread Strength (N)'),...
   title('Relative Frequency Histogram for 100 Tests')
```

The result appears in Figure 7.1–3.

The fourth, fifth, and sixth forms of the `hist` function do not generate a plot, but are used to compute the frequency counts and bin locations. The `bar` function can then be used to plot the histogram. The syntax of the fourth form is `[z,x] = hist(y)`, where `z` is the returned vector containing the frequency count and `x` is the returned vector containing the bin locations. The fifth and sixth forms are `[z,x] = hist(y,n)` and `[z,x] = hist(y,x)`. In the latter case the returned vector `x` is the same as the user-supplied vector. The following script file shows how the sixth form can be used to generate a relative frequency histogram for the thread example with 100 tests.

```
tests = 100;
y = [91*ones(1,13),92*ones(1,15),93*ones(1,22),...
   94*ones(1,19),95*ones(1,17),96*ones(1,14);
```

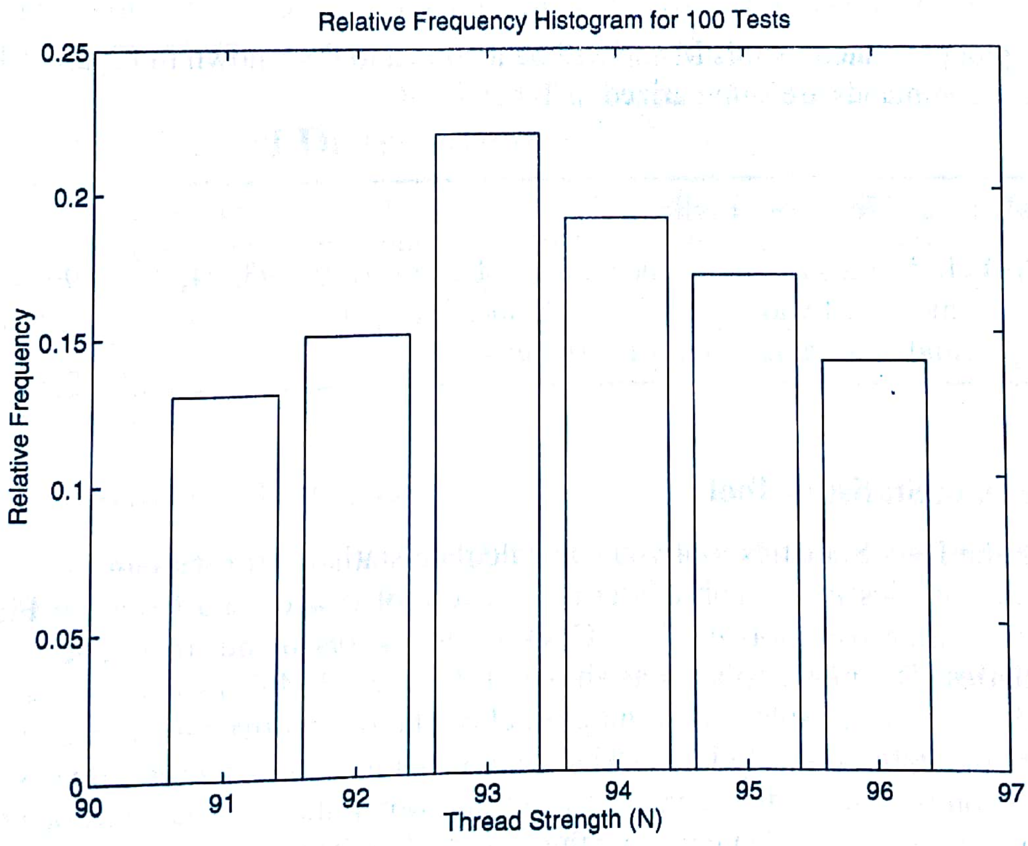


Figure 7.1–3 Relative frequency histogram for 100 thread tests.

Table 7.1–1 Histogram functions

Command	Description
`bar(x,y)`	Creates a bar chart of `y` versus `x`.
`hist(y)`	Aggregates the data in the vector `y` into 10 bins evenly spaced between the minimum and maximum values in `y`.
`hist(y,n)`	Aggregates the data in the vector `y` into `n` bins evenly spaced between the minimum and maximum values in `y`.
`hist(y,x)`	Aggregates the data in the vector `y` into bins whose center locations are specified by the vector `x`. The bin widths are the distances between the centers.
`[z,x] = hist(y)`	Same as `hist(y)` but returns two vectors `z` and `x` that contain the frequency count and the bin locations.
`[z,x] = hist(y,n)`	Same as `hist(y,n)` but returns two vectors `z` and `x` that contain the frequency count and the bin locations.
`[z,x] = hist(y,x)`	Same as `hist(y,x)` but returns two vectors `z` and `x` that contain the frequency count and the bin locations. The returned vector `x` is the same as the user-supplied vector `x`.

```
x = 91:96;
[z,x] = hist(y,x);bar(x,z/tests),...
   ylabel('Relative Frequency'),xlabel('Thread Strength(N)'),...
   title('Relative Frequency Histogram for 100 Tests')
```

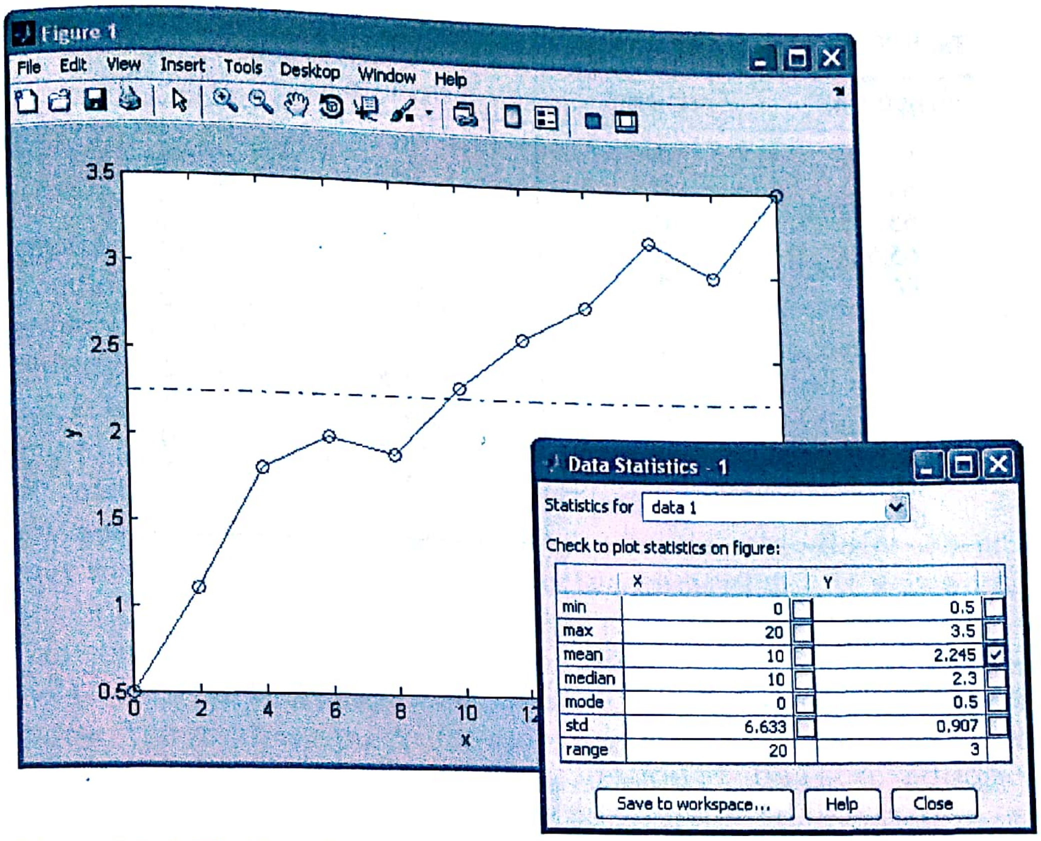
The plot generated by this M-file will be identical to that shown in Figure 7.1–3. These commands are summarized in Table 7.1–1.

Test Your Understanding

T7.1–1 In 50 tests of thread, the number of times 91, 92, 93, 94, 95, or 96 N was measured was 7, 8, 10, 6, 12, and 7, respectively. Obtain the absolute and relative frequency histograms.

The Data Statistics Tool

With the Data Statistics tool you can calculate statistics for data and add plots of the statistics to a graph of the data. The tool is accessed from the Figure window after you plot the data. Click on the **Tools** menu, then select **Data Statistics.** The menu appears as shown in Figure 7.1–4. To show the mean of the dependent variable (y) on the plot, click the box in the row labeled `mean` under the column labeled `Y`, as shown in the figure. A horizontal line is then placed on the plot at the mean. You can plot other statistics as well; these are shown in the figure. You can save the statistics to the workspace as a structure by clicking on the **Save to Workspace** button. This opens a dialog box that



Data Statistics - 1

Statistics for data 1

Check to plot statistics on figure:

	X		Y	
min	0	☐	0.5	☐
max	20	☐	3.5	☐
mean	10	☐	2.245	✓
median	10	☐	2.3	☐
mode	0	☐	0.5	☐
std	6.633	☐	0.907	☐
range	20		3	

Figure 7.1–4 The Data Statistics tool.

prompts you for a name for the structure containing the *x* data, and a name for the *y* data structure.

7.2 The Normal Distribution

Rolling a die is an example of a process whose possible outcomes are a limited set of numbers, namely, the integers from 1 to 6. For such processes the probability is a function of a discrete-valued variable, that is, a variable having a limited number of values. For example, Table 7.2–1 gives the measured heights of 100 men 20 years of age. The heights were recorded to the nearest 1/2 in., so the height variable is discrete-valued.

Scaled Frequency Histogram

You can plot the data as a histogram using either the absolute or relative frequencies. However, another useful histogram uses data scaled so that the total area under the histogram's rectangles is 1. This *scaled frequency histogram* is the absolute frequency histogram divided by the total area of that histogram. The area of each rectangle on the absolute frequency histogram equals the bin width times the absolute frequency for that bin. Because all the rectangles have the same width, the total area is the bin width times the sum of the absolute frequencies. The following M-file produces the scaled histogram shown in Figure 7.2–1.

Table 7.2–1 Height data for men 20 years of age

Height (in.)	Frequency	Height (in.)	Frequency
64	1	70	9
64.5	0	70.5	8
65	0	71	7
65.5	0	71.5	5
66	2	72	4
66.5	4	72.5	4
67	5	73	3
67.5	4	73.5	1
68	8	74	1
68.5	11	74.5	0
69	12	75	1
69.5	10		

```
% Absolute frequency data.
y_abs=[1,0,0,0,2,4,5,4,8,11,12,10,9,8,7,5,4,4,3,1,1,0,1];
binwidth = 0.5;
% Compute scaled frequency data.
area = binwidth*sum(y_abs);
y_scaled = y_abs/area;
```

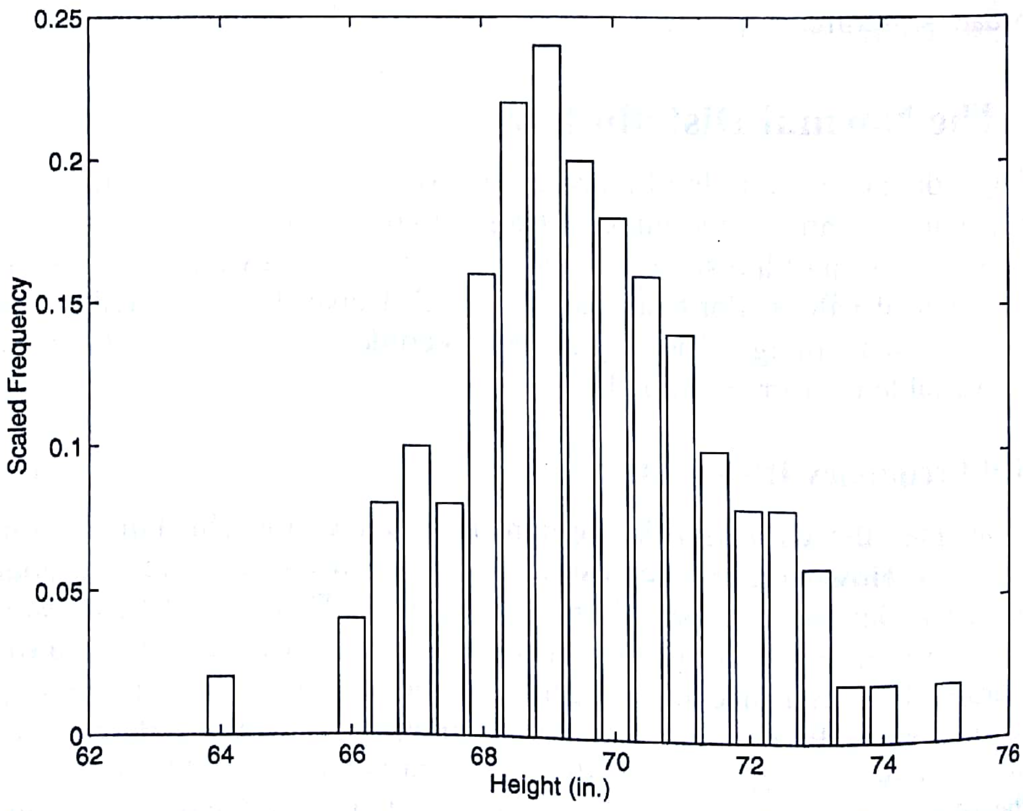


Figure 7.2–1 Scaled histogram of height data.

```
% Define the bins.
bins = 64:binwidth:75;
% Plot the scaled histogram.
bar(bins,y_scaled),...
   ylabel('Scaled Frequency'),xlabel('Height (in.)')
```

Because the total area under the scaled histogram is 1, the fractional area corresponding to a range of heights gives the probability that a randomly selected 20-year-old man will have a height in that range. For example, the heights of the scaled histogram rectangles corresponding to heights of 67 through 69 in. are 0.1, 0.08, 0.16, 0.22, and 0.24. Because the bin width is 0.5, the total area corresponding to these rectangles is (0.1+ 0.08 + 0.16 + 0.22 + 0.24)(0.5) = 0.4. Thus 40 percent of the heights lie between 67 and 69 in.

You can use the `cumsum` function to calculate areas under the scaled frequency histogram and therefore to calculate probabilities. If `x` is a vector, `cumsum(x)` returns a vector the same length as `x`, whose elements are the sum of the previous elements. For example, if `x =[2, 5, 3, 8]`, `cumsum(x)= [2, 7, 10, 18]`. If `A` is a matrix, `cumsum(A)` computes the cumulative sum of each row. The result is a matrix the same size as `A`.

After running the previous script, the last element of `cumsum(y_scaled)* binwidth` is 1, which is the area under the scaled frequency histogram. To compute the probability of a height lying between 67 and 69 in. (that is, above the 6th value up to the 11th value), type

```
>>prob = cumsum(y_scaled)*binwidth;
>>prob67_69 = prob(11)-prob(6)
```

The result is `prob67_69 = 0.4000`, which agrees with our previous calculation of 40 percent.

Continuous Approximation to the Scaled Histogram

For processes having an infinite number of possible outcomes, the probability is a function of a *continuous* variable and is plotted as a curve rather than as rectangles. It is based on the same concept as the scaled histogram; that is, the total area under the curve is 1, and the fractional area gives the probability of occurrence of a specific range of outcomes. A probability function that describes many processes is the *normal* or *Gaussian* function, which is shown in Figure 7.2–2.

NORMAL OR GAUSSIAN FUNCTION

This function is also known as the *bell-shaped curve*. Outcomes that can be described by this function are said to be *normally distributed*. The normal probability function is a two-parameter function; one parameter, μ, is the mean of the outcomes, and the other parameter, σ, is the *standard deviation*. The mean μ locates the peak of the curve and is the most likely value to occur. The width, or spread, of the curve is described by the parameter σ. Sometimes the term *variance* is used to describe the spread of the curve. The variance is the square of the standard deviation σ.

NORMALLY DISTRIBUTED

STANDARD DEVIATION

VARIANCE

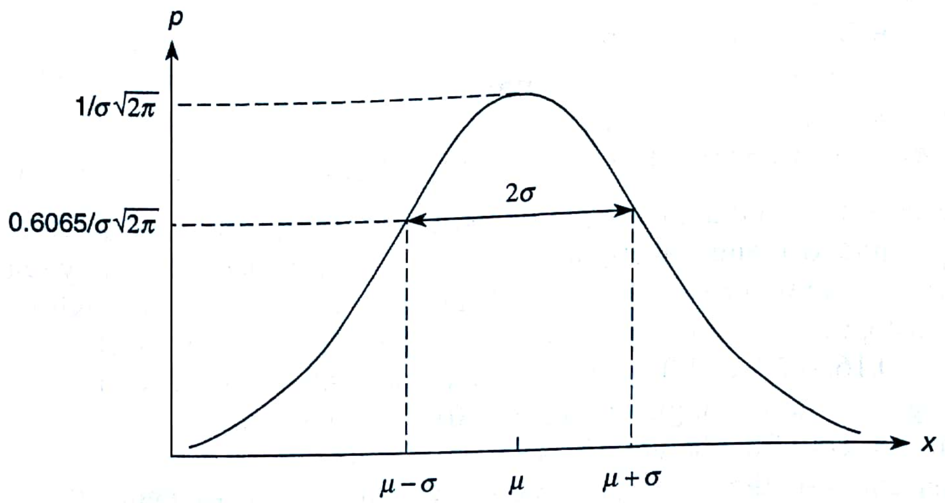


Figure 7.2–2 The basic shape of the normal distribution curve.

The normal probability function is described by the following equation:

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2} \tag{7.2–1}$$

It can be shown that approximately 68 percent of the area lies between the limits of $\mu - \sigma \leq x \leq \mu + \sigma$. Consequently, if a variable is normally distributed, there is a 68 percent chance that a randomly selected sample will lie within one standard deviation of the mean. In addition, approximately 96 percent of the area lies between the limits of $\mu - 2\sigma \leq x \leq \mu + 2\sigma$, and 99.7 percent, or practically 100 percent, of the area lies between the limits of $\mu - 3\sigma \leq x \leq \mu + 3\sigma$.

The functions `mean(x)`, `var(x)`, and `std(x)` compute the mean, variance, and standard deviation of the elements in the vector `x`.