1

DATA

Arguably, the most important role of the applied mathematical scientist is to model and analyze data. While this statement may appear to be obvious, just how to proceed given any set of data remains as much art as it does science. Should the data be calibrated, normalized, transformed, smoothed, filtered, separated into subclasses? These are just a few of the approaches one can take to data before beginning any substantial analysis. Therefore, this chapter will be dedicated to a select few methods: Data visualization, data transformation, data filtering, data clustering (not to be confused with clustered data of Chapter 4), and data quality.

1.1 DATA VISUALIZATION

Yogi Berra, the Baseball Hall of Fame catcher for the New York Yankees 1946–1965 and accidental metaphysician once remarked [2; https://en.wikipedia.org/wiki/Yogi_Berra]

You can observe a lot by watching.

Fanciful or not, this comment takes to heart the power of modern computing and the MATLAB software system in particular. In the Glossary of MATLAB Functions written for this text, the reader will find a number of specialized visualization M-files designed to plot data that have been processed or transformed. This includes

Applied Mathematics for the Analysis of Biomedical Data: Models, Methods, and MATLAB®, First Edition. Peter J. Costa.

Companion website: www.wiley.com/go/costa/appmaths_biomedical_data

principal component axes and *discriminant analysis feature extraction coordinates* for multivariate data. For single variable matched-pairs data, *Bland–Altman plots* of *allowable total difference zones* can be constructed.

More practically, the question this section will focus on is "What can be inferred or deduced from these data?"

Consider a question raised in Chapter 5. That is, height, weight, and age data for every 2014–2015 season active National Basketball Association (NBA) and National Hockey League (NHL) player are available from www.nba.com and www.nhl.com, respectively. For the sake of convenience, these data are summarized in two separate MAT files contained in the software associated with this text. How do the heights of the players from both leagues compare? Figures 5.1a and 5.1b of Chapter 5 provide the weighted histograms for the player height data. This is certainly one way to examine these data. Alternately, a point-by-point plot of the data along with a means of each collection of measurements is presented in Figure 1.1. The dots (•) on the upper graph represent the heights of the active 2014–2015 NHL players and the thick line (_____) through the center of these data is the average height (indicated by the symbol μ_{NHL}). In a similar manner, the dots (•) on the lower portion of the graph represent the active 2014–2015 NBA players with average height (_____) denoted by μ_{NBA}. Instinctively, the sense of the figure is that (on average) NBA players are taller than NHL players. This is verified in Chapter 5 on hypothesis testing.

While the *average* player heights are statistically different (see Table 5.1 of Chapter 5), the question of *classification* remains. More specifically, from the triple of height, weight, and age, can such information be used to determine whether a particular player measurement (*Ht*, *Wt*, *Age*) indicates to which *class* (namely, NBA or

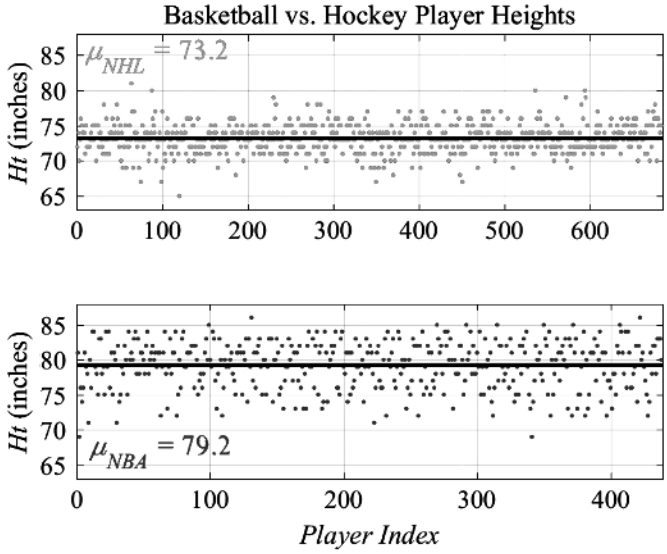

FIGURE 1.1 Professional athletes' height data.

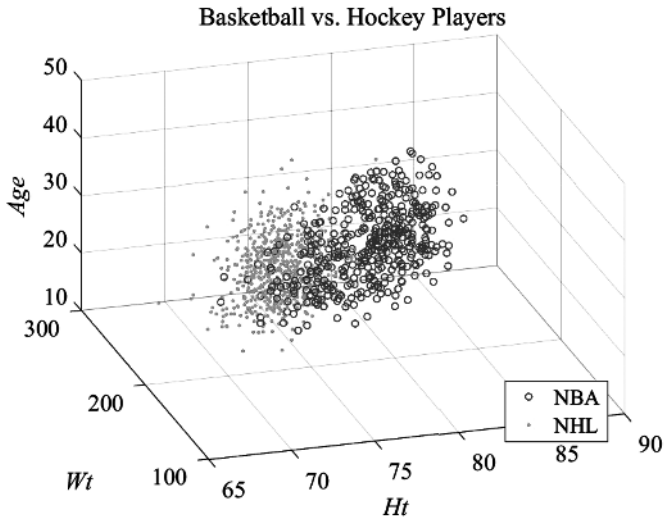

FIGURE 1.2 Height, weight, age data for NBA and NHL players.

NHL) the player belongs? The answer is discussed in Chapter 4 on classification. The raw data, obtained from www.nhl.com and www.nba.com, for the active players in the 2014–2015 season can be viewed in Figure 1.2. What is the best approach to be used in separating these groups? This will be the concern of the next section.

EXERCISES

1.1 Make plots of the NBA vs. NHL weight data. Do the same for the *age* data. Are the differences plain from the graphs? These data are located in `C:\Database\Math_Biology\Chapter_1\Data` as `BasketballData.mat` and `HockeyData.mat`. They are obtained via the MATLAB commands

```
Ddir = 'C:\Database\Math_Biology\Chapter_1\Data';
load(fullfile(Ddir,'BasketballData.mat'));
load(fullfile(Ddir,'HockeyData.mat'));
```

Hint: The weight data are `B.Wt` and `D.Wt` while the age data are `B.Ages` and `D.Ages`.

1.2 DATA TRANSFORMATIONS

Once data have been recorded, how should they be treated so that the *content* of the information contained therein is most plainly revealed? This question is unanswerable, as it presupposes there is any discriminatory information within the measurements. There are, however, standard approaches that can be applied to make the data more regular. These methods are listed sequentially.

1.2.1 Normalization

The basic idea behind *normalization* is to collect a representative data set and then, from each measurement within the set, subtract the mean and divide by the standard deviation. If the data are multidimensional, this can be achieved by *subtracting off* the mean vector and multiplying by the inverse of the associated covariance matrix. This notion is made precise in Chapter 4, equation (4.5). As a review, suppose that $X \in \mathcal{Mat}_{n\times p}(\mathbb{R})$ is the data matrix of n p-dimensional measurements from the same source. For the NHL (*Ht*, *Wt*, *Age*) data, this means that $n = 684$ and $p = 3$. The *standard normalization* of the data matrix X is

$$Z = (X - \mathbf{1}_{n\times 1} \cdot \boldsymbol{m}) \cdot S^{-1}. \tag{1.1}$$

Here $\mathbf{1}_{n\times 1} = [1, 1,\ldots, 1]^T$ is the n-dimensional *column vector* of 1's, $\boldsymbol{m} = [m_1, m_2,\ldots, m_p]$ is the p-dimensional row vector whose every element is the column mean of the data matrix X. Finally, $S = diag(s_1, s_2,\ldots, s_p)$ is the $p \times p$ diagonal matrix whose nonzero entries are the column standard deviations of X (e.g., see Johnson and Wichern [8] for details about normalization). Applying this transformation to the data matrices for the (*Ht*, *Wt*, *Age*) NBA and NHL triplets results in the normalized data displayed in Figure 1.3.

The reader can see that the normalized data provide a comparable display to the unnormalized data.

Rather than simply normalizing the data, projecting these measurements into coordinates that amplify differences is now prescribed.

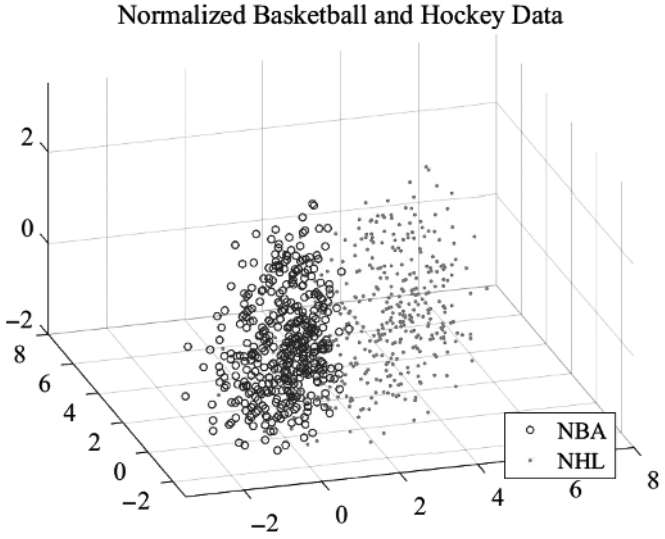

FIGURE 1.3 Normalized NBA and NHL player (*Ht*, *Wt*, *Age*) data.

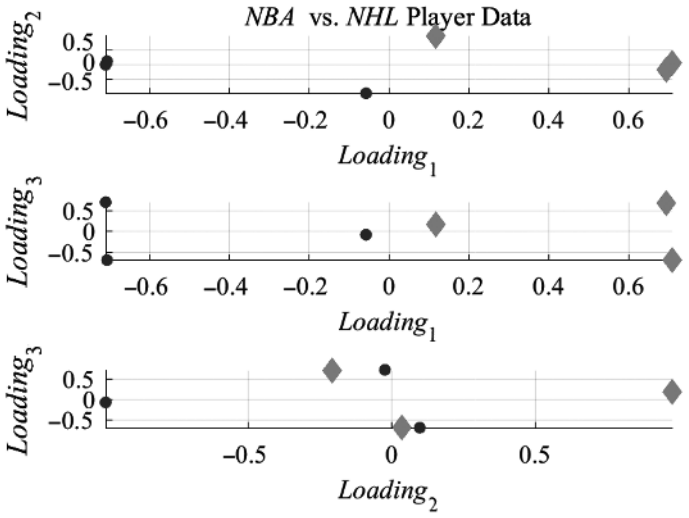

FIGURE 1.4 Loadings for the NBA • and NHL ◆ (*Ht*, *Wt*, *Age*) data.

1.2.2 Principal Components (Karhunen–Loève Transform)

In Section 4.3 (Chapter 4), the notion of principal components as an orthogonal set of axes along the descending amounts of variation in a data matrix X is detailed. In particular, equations (4.9a)–(4.10d) present the principal component axes and projection into *PC space*. The projection, defined in Chapter 4, equation (4.10d) is sometimes referred to as the *Karhunen–Loève* transform. As noted in Chapter 4, the projection matrix is a product of the first ρ singular values[1] of the data matrix X and the associated loading matrix. Here, ρ is the desired number of principal components (e.g., the number of *PCs* required to attain say 99.9% of the total data variation). Figure 1.4 illustrates the loadings for the NBA/NHL (*Ht*, *Wt*, *Age*) data analysis.

To see *how many PCs* are required to meet a desired percentage of the total variance, a *screeplot* can be created. This is a depiction of the percentage of the total variance captured by the number of principal components used as a function of the normalized eigenvalues.

The data visualized in this graphic are taken from a four class, multivariate collection of measurements in which one level of a disease is normal (no-disease) and the remaining three are of increasingly serious levels. The screeplot in Figure 1.5 gives the following information. For levels 0–2, it is seen that 5–6 *PCs* provide the vast majority of the variance information. At level 3, there are only 10 non-trivial *PCs* with the bulk of the information contained in the first 5 *PCs*. The data have been normalized as per (1.1).

[1] For more information about linear algebra and singular values, see Appendix Section A.2.

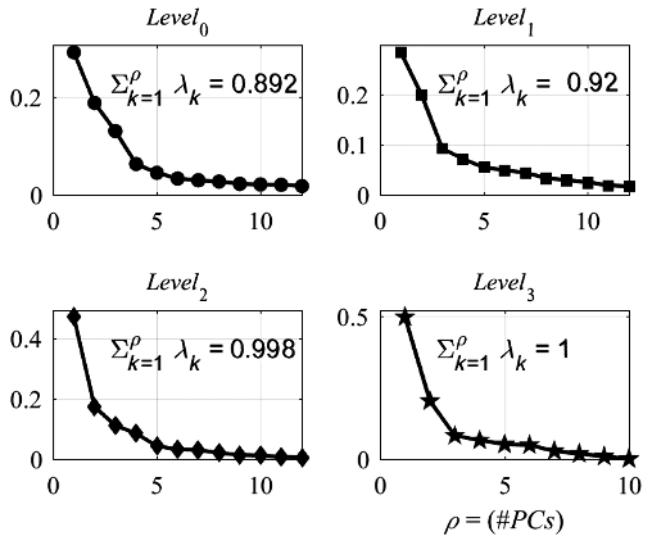

FIGURE 1.5 Screeplot for up to 12 *PCs* for multivariate data.

1.2.3 DAFE Coordinates

Discriminant analysis feature extraction (DAFE) coordinates are independent axes along the direction of maximal discriminant information. The DAFE axes projection mapping Π_r into the first *r DAFE coordinates* is composed of the first r columns of the orthogonal matrix from the singular value decomposition of the Fisher discriminant matrix. This matrix $F = C_{pool}^{-1} \cdot C_{btwn}$ is described in equation (4.13b) of Section 4.4 (Chapter 4). The projection matrix Π_r is defined via equation (4.10d) of Section 4.3 (Chapter 4). Figures 4.10 and 4.11 of Chapter 4 illustrate DAFE coordinates and the corresponding weightings on each class projection.

The exercises will give the reader some experience in reduction of dimension via DAFE coordinates and how such axes act to separate data classes.

EXERCISES

1.2 Figure 1.4 illustrates the PCA loadings for the NBA vs. NHL (*Ht*, *Wt*, *Age*) data. Use the M-file `scorecompare.m` to plot the PCA scores for these data. These data are obtained via the MATLAB commands

```
Ddir = 'C:\Database\Math_Biology\Chapter_1\Data';
load(fullfile(Ddir,'BasketballData.mat'));
load(fullfile(Ddir,'HockeyData.mat'));
```

1.3 How many DAFE coordinates are required to recover 99.9% of the discriminant information contained in the NBA vs. NHL (*Ht*, *Wt*, *Age*) data? Use the M-files `dafe.m` and `dafeplot.m` and the commands in Exercise 1.2 to project the data into DAFE space and then plot it, respectively.

1.3 DATA FILTERING

Data are by their very nature noisy. Indeed, data are measurements that are recorded either via device or human beings. No machine can perfectly register a series of measurements without some manner of error. And we humans are legendary for our inability to accurately record and repeat measurements. It is this portion of our humanity that motivates the design and development of machines to perform repetitive tasks. Consequently, any set of measurements must be viewed as inherently imprecise.

Consider the simulated respiratory infection data from Chapter 3 (Figure 3.10b). While these data follow what appears to be a regular sinusoidal pattern, it is evident that there is plenty of "jitter" in the plot. One approach to filter the data is to smooth the signal. This can be achieved in a number of ways. Two methods will be discussed: Convolution and Fourier transforms.

1.3.1 Convolution and Smoothing

Again, referring to the respiratory infections data of Chapter 3, the question of how to smooth this signal arises. Smoothing by convolution is one approach. If $f(t)$ is an integrable function, then the *convolution* of f with the function g is defined as

$$(f \odot g)(t) = \int_{-\infty}^{\infty} f(s) \cdot g(t-s)\, ds. \tag{1.2}$$

If $g(t)$ is a "square wave" as indicated in Figure 1.6, then by convolving against the function f, the nonzero portion of g smooths those portions of f of equal length

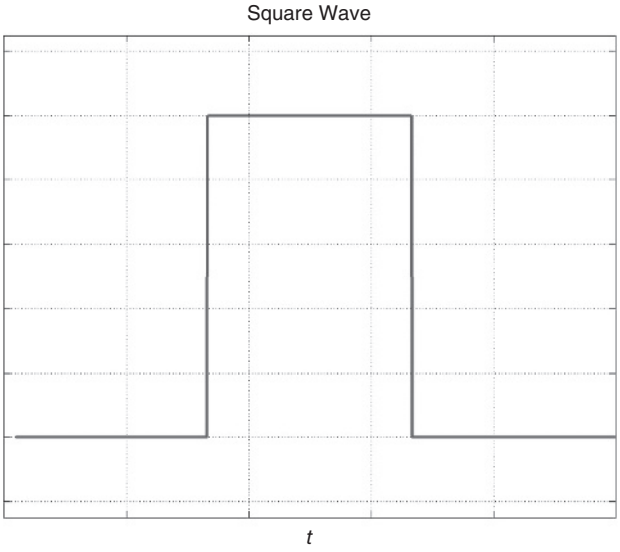

FIGURE 1.6 Square wave smoothing function.

to g. Andrews and Shivamoggi [1] and Costa [3] contain discussions on convolution and Fourier transforms (which will also be used to smooth signals).

The numerical algorithm used for smoothing a signal $f(t)$ via a smoothing function $g(t)$ is presented in the next section. For the sake of simplicity, it will be assumed that the nonzero portion of g consists of 2^p points.

1.3.1.1 Convolution Smoothing Algorithm

(i) Convolve f with g via (1.2). Obtain $\phi(t) = (f \odot g)(t)$.

(ii) Select the smoothing function g to have 2^p elements. Remove the first $2^{p-1} - 1$ and the last $2^{p-1} - 1$ elements from the new convolution vector $\phi(t)$.

(iii) Normalize $\phi(t)$ by the number of nonzero points in the smoothing function g.

Figure 1.7 illustrates the results of this smoothing approach. Observe that as the number of elements in the smoothing vector increases so does the amount of smoothing. As the amount of smoothing increases, however, the results exhibit two deficiencies: Loss of amplitude and increase in "ghosting." Ghosting or *aliasing* is the artificial suppression of the smoothed data near the initial and terminal portions of the signal. The first (loss of amplitude) is evident from the plot. At the end of the smoothed functions, moreover, tails trail off to zero. This behavior largely ignores the original data trend. This phenomenon is called *aliasing* (see, e.g., Costa [3] for more details). The MATLAB code used to generate the smoothed functions displayed in Figure 1.7 is presented below.

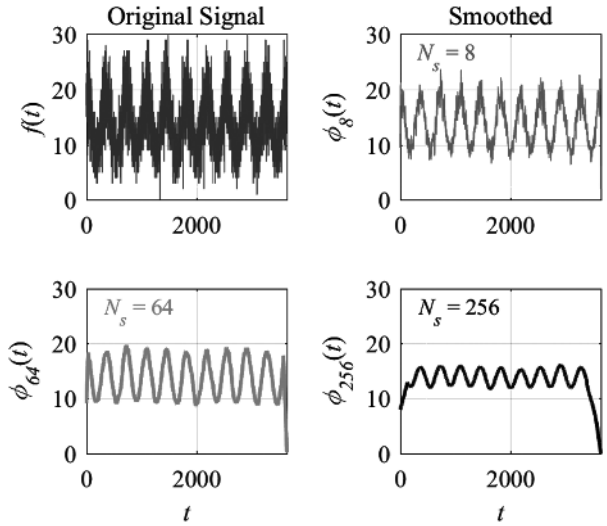

FIGURE 1.7 Smoothing for $N_s = 8$ (top right), 64 (lower left), and 256 (lower right) elements.

MATLAB Commands
(*Smoothing via Convolution*)

```
% Data directory …
Ddir = 'C:\Database\Math_Biology\Chapter_1\Data';
% … for the respiratory infection data.
data = load(fullfile(Ddir,'SEIR_Simulated_Data.mat'));
t = load(fullfile(Ddir,'SEIR_Time.mat'));
% Place the data and time into MATLAB vectors
D = data.D; T = t.T; clear data t
% Number of measurements (infected patients)
N = numel(D);
% Smoothing vectors of various lengths
s{1} = ones(1,2^3); s{2} = ones(1,2^6); s{3} = ones(1,2^8);
% Number of elements per smoothing vector
Ns = cellfun(@numel,s);

% Convolve the data with the smoothing vectors s
for j = 1:numel(s);
  % indices to be removed from the right and left portion of the convolution
  iR = [1:(-1+Ns(j)/2),N-(-1+Ns(j)/2):N];
  % Convolve D and s: (D ⊙ s)(t)
  tmp = conv(D,s{j});
  % Remove the points at the undesired indices
  tmp(iR) = [];
  % Normalize the convolution
  Ds{j} = tmp/Ns(j);
end
```

1.3.2 Fourier Transform and Smoothing

Rather than using convolution to smooth a signal, the *Fourier transform* can be utilized. Definitions for the transform and its inverse are provided in equations (1.3a) and (1.3b). The basic idea is that the Fourier transform maps a time (or space)-based function into frequency space. By eliminating high frequencies in the transformed function, the inverse transform will smooth the frequency-truncated data. This is formalized via the algorithm and Figure 1.8.

$$\mathcal{F}[f(t)](\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t)\, e^{-it\omega}\, dt \tag{1.3a}$$

$$\mathcal{F}^{-1}[F(\omega)](t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} F(\omega)\, e^{it\omega}\, d\omega \tag{1.3b}$$

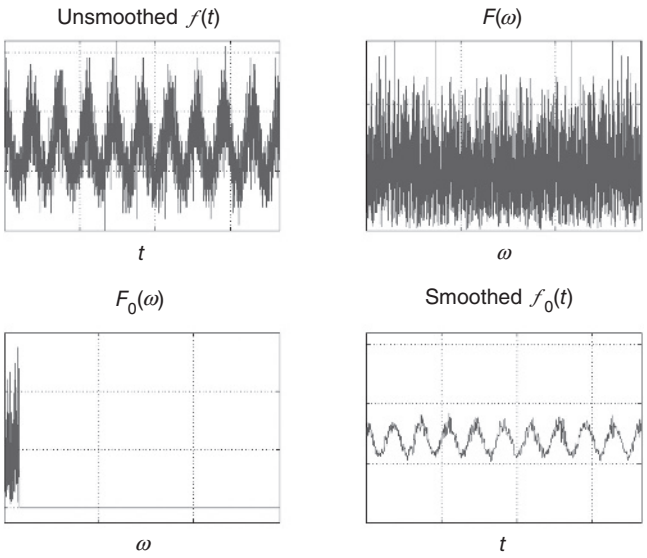

FIGURE 1.8 Smoothing via Fourier transform.

Remarks:

(1) The notation $F(\omega)$ is often used in place of $\mathfrak{F}[f(t)](\omega)$. Also, the transform and its inverse are frequently defined in the alternate forms $F(\omega) = \int_{-\infty}^{\infty} f(t)\, e^{-2\pi i t\omega}\, dt$ and $f(t) = \int_{-\infty}^{\infty} F(\omega)\, e^{2\pi i t\omega}\, d\omega$.

(2) The *Heaviside* function is the distribution whose values are 1 on the positive real line. That is, $H(x) = \begin{cases} 1 & \text{for } x \geq 0 \\ 0 & \text{otherwise} \end{cases}$. Thus, $S(x,\ 1) = H(x) - H(x - 1)$ is a *square wave*. More generally, $S(x,\ x_0) = H(x) - H(x - x_0)$ is the rectangular wave defined over $[0,\ x_0]$ and $S(x, [x_a, x_b]) = H(x - x_a) - H(x - x_b)$ is defined over $[x_a, x_b]$.

1.3.2.1 Smoothing Algorithm Fourier Transform

(i) Apply the Fourier transform to the function $f(t)$ and obtain $F(\omega)$.

(ii) Multiply $F(\omega)$ by $S(\omega,\ \omega_0)$ to eliminate any frequencies greater than ω_0. $F_0(\omega) = F(\omega) \cdot S(\omega,\ \omega_0)$.

(iii) Apply the inverse Fourier transform to $F_0(\omega)$.

(iv) Obtain the smoothed function $f_0(t) = \mathfrak{F}^{-1}[F_0(\omega)](t)$.

These steps are illustrated in Figure 1.8 as upper left (step i), upper right (step ii), lower left (step iii), and lower right (step iv). Since the function $f(t)$ is obtained as

the vector of measurements $\boldsymbol{f} = [f_1, f_2, \ldots, f_n]$, the *discrete Fourier transform* (DFT) is used. The DFT and its inverse are provided via equations (1.4a) and (1.4b).

$$DFT[f] = \left[\sum_{j=1}^{n} f_j, \sum_{j=1}^{n} f_j (e^{-2\pi i \cdot (j-1)}), \sum_{j=1}^{n} f_j (e^{-2\pi i \cdot 2(j-1)}), \cdots, \sum_{j=1}^{n} f_j (e^{-2\pi i \cdot (n-1) \cdot (j-1)}) \right] \tag{1.4a}$$

$$IDFT[F] = \frac{1}{n} \left[\sum_{j=1}^{n} F_j, \sum_{j=1}^{n} F_j (e^{2\pi i \cdot (j-1)}), \sum_{j=1}^{n} F_j (e^{2\pi i \cdot 2(j-1)}), \cdots, \sum_{j=1}^{n} F_j (e^{2\pi i \cdot (n-1) \cdot (j-1)}) \right] \tag{1.4b}$$

As with the smoothing via convolution, the greater the amount of smoothing, the smaller the amplitude envelop on the smoothed function. The *Fourier* method does not appear to present the problem of aliasing. The MATLAB code below is used to compute the smoothed vector $\boldsymbol{f}_0$.

MATLAB Commands
(*Smoothing via Fourier Transform*)

```
% Data directory …
Ddir = 'C:\Database\Math_Biology\Chapter_1\Data';
% … for the respiratory infection data.
data = load(fullfile(Ddir,'SEIR_Simulated_Data.mat'));
t = load(fullfile(Ddir,'SEIR_Time.mat'));
% Place the data and time into MATLAB vectors
D = data.D; T = t.T; clear data t
% Number of measurements (infected patients)
N = numel(D);
% Take the (discrete) Fourier of the Respiratory data
F = fft(D);
% Form the square wave by removing all but the first 200 wavelengths.
S = ones(1,N); w = T/(2*pi); wo = w(200);
ind = w > wo; S(ind) = 0;
% Compute the Fourier transform of the truncated frequency data and its inverse
transform
Fo = F.*S; fo = ifft(Fo);
% Plot the results step-by-step
figure;
% Unsmoothed function f(t)
subplot(2,2,1); plot(T,D,'b'); grid('on');
set(gca,'FontSize',16,'FontName','Times New Roman');
title('Unsmoothed \itf\rm(\itt\rm)');
axis([0,T(end),0,max(ceil(D))]);
% Discrete Fourier transform of f(t), F(ω)
```

```
subplot(2,2,2); plot(w,abs(F),'b'); grid('on');
set(gca,'FontSize',16,'FontName','Times New Roman');
title('\itF\rm(\it\omega\rm)'); axis([0,w(end),0,600]);
% Truncated wavelength transform F_0(ω)
subplot(2,2,3); plot(w,abs(Fo),'b'); grid('on');
set(gca,'FontSize',16,'FontName','Times New Roman');
title('\itF_{o}\rm(\it\omega\rm)'); axis([0,w(end),-50,600]);
% Inverse transform of F_0(ω) ⇒ smoothed function f_0(t)
subplot(2,2,4); plot(T,abs(fo),'b'); grid('on');
set(gca,'FontSize',16,'FontName','Times New Roman');
title('Smoothed \itf_{o}\rm(\itt\rm)'); axis([0,T(end),0,
max(ceil(D))]);
```

1.3.3 Outlier Filtering

Smoothing alters *every* element of the data set under consideration. What happens if there are only a few "extraordinary" members of the collection of measurements? How can such exceptional measurements be identified?

The first course of action is to define what "extraordinary" or "exceptional" means. Such measurements are referred to as *outliers*. An *outlier* is a measurement that is markedly different from all others in a sample. The notion of an outlier is simultaneously obvious and difficult to formalize. Indeed, outliers can best be described by a phrase made famous by the US Supreme Court Associate Justice Potter Stewart [12].

> *I shall not today attempt further to define the kinds of material I understand to be embraced within that shorthand description; and perhaps I could never succeed in doing so. But I know it when I see it.*

To enhance the presentation, let $\boldsymbol{x} = \{x_1, x_2, \ldots, x_n\}$ be a sample of a random variable X. One approach to identify sample outliers is to normalize each element of the sample by the mean and standard deviation. That is, set $z_j = \dfrac{x_j - \bar{x}}{s}$ for $j = 1, 2, \ldots, n$, $\bar{x}$ = the sample mean, and s = the sample standard deviation. The z_j comprise the *normalized sample* and, in limit, follow a standard normal distribution (for more information concerning probability distributions, see Appendix Section A.3). Consequently, measurements at the extreme margins (e.g., the 0.1% or 99.9% quantiles) can be considered *outliers*. If ω is the selected quantile, then $\mathcal{I}(\omega) = [\bar{x} - s \cdot \omega, \bar{x} + s \cdot \omega]$ is the *inclusion interval*. Thus, $x_k \notin \mathcal{I}(\omega)$ means that x_k is an *outlier*. This method can be reviewed in Johnson and Wichern [8] and with limiting values suggested by Tholen et al. [13]. For non-symmetric quantile limits (i.e., the lower limit is a 0.5% quantile while the upper limit is the 99.99% quantile), the inclusion interval can be generalized to $\mathcal{I}(\boldsymbol{\omega}) = [\bar{x} - s \cdot \omega_\ell, \bar{x} + s \cdot \omega_u]$, where $\boldsymbol{\omega} = [\omega_\ell, \omega_u]$ is the vector of the lower and upper quantiles ω_ℓ and ω_u, respectively.

This section will provide a different method for identifying outliers. The details of this development can be viewed in Costa [4].

Rather than normalizing the sample as above, replace the minimum and maximum of the sample by its nearest neighbor. Specifically, let $\boldsymbol{x}_{order} = \{x_{(1)}, x_{(2)},\ldots, x_{(n)}\}$ be the ordered values of $\boldsymbol{x}$. That is, $\min_{1\le j\le n}\{x_j\} \equiv x_{(1)} \le x_{(2)} \le x_{(3)} \le \cdots \le x_{(n)} \equiv \max_{1\le j\le n}\{x_j\}$. Next remove $x_{(1)}$ and $x_{(n)}$ from the sample and replace them by $x_{(2)}$ and $x_{(n-1)}$, respectively. The *truncated* sample $\boldsymbol{x}_T = \{x_{(2)}, x_{(2)}, x_{(3)}, x_{(4)},\ldots, x_{(n-1),}\, x_{(n-1)}\}$ is then used to normalize the sample via the truncated mean $\bar{x}_T$ and standard deviation s_T.

$$z_j = \frac{x_j - \bar{x}_T}{s_T} \tag{1.5a}$$

$$\bar{x}_T = \frac{1}{n}\left(x_{(2)} + \sum_{j=2}^{n-1} x_{(j)} + x_{(n-1)}\right) \tag{1.5b}$$

$$s_T = \sqrt{\frac{1}{n-1}\left((x_{(2)} - \bar{x}_T)^2 + \sum_{j=2}^{n-1}(x_{(j)} - \bar{x}_T)^2 + (x_{(n-1)} - \bar{x}_T)^2\right)} \tag{1.5c}$$

For a selected quantile ω the *truncated inclusion interval* is

$$\mathcal{I}_T(\omega) = [\bar{x}_T - s_T \cdot \omega, \bar{x}_T + s_T \cdot \omega] \tag{1.5d}$$

If $x_k \notin \mathcal{I}_T(\omega)$, then x_k is an outlier with respect to the *truncated outlier filtering method*. To see how the truncated outlier filter contrasts with conventional outlier identification, consider a sample formed by selecting 100 draws from a uniform distribution over [0, 1] and two draws from $\mathcal{U}[0, 10]$. That is, $\boldsymbol{x} = \{x_1, x_2, x_3,\ldots, x_{102}\}$ where $x_{i_1}, x_{i_2}, \ldots, x_{i_{100}} \overset{i.i.d}{\sim} \mathcal{U}[0, 1]$ and $x_{i_{101}}, x_{i_{102}} \overset{i.i.d}{\sim} \mathcal{U}[0, 10]$ for some set of indices $\{i_1, i_2,\ldots, i_{102}\} \subset \{1, 2,\ldots, 102\}$. The conventional outlier detection method normalizes the data with respect to the sample mean and sample standard deviation taken with respect to the entire data set. The inclusion interval is then calculated with respect to a selected quantile ω. The truncated outlier filter uses equations (1.5a)–(1.5d) to calculate the inclusion interval. Figure 1.9 provides an illustration of conventional versus truncated outlier filtering. As can be seen, the conventional method identifies only one of the samples from $\mathcal{U}[0, 10]$ as an outlier while the truncated method indicates that both samples from $\mathcal{U}[0, 10]$ are outliers. Moreover, the inclusion interval produced by the conventional method (illustrated along the y-axis) is considerably wider than the truncated filter method.

Among the virtues of the truncated outlier filter is its extensibility to higher dimensions. Indeed, rather than n one-dimensional samples $\boldsymbol{x} = \{x_1, x_2,\ldots, x_n\}$, suppose there are n p-dimensional samples contained in the data matrix X.

$$X = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,p} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,p} \end{bmatrix} \equiv \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_n \end{bmatrix} \tag{1.6}$$

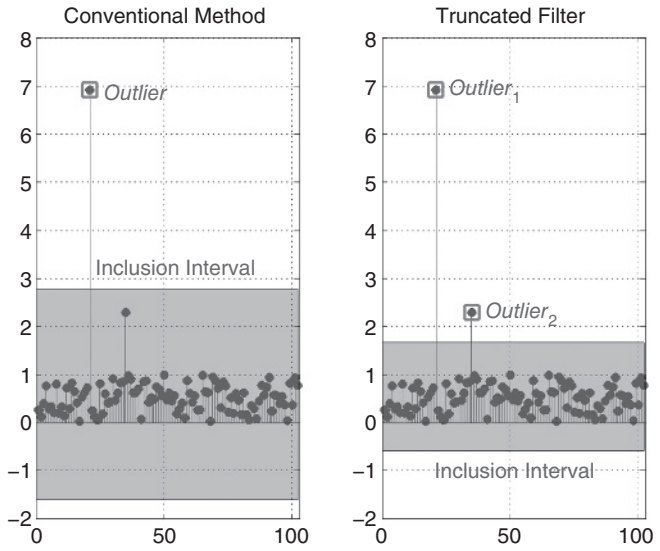

FIGURE 1.9 Outliers identified via conventional and truncated filter methods.

Now, order the data matrix column-wise to obtain

$$X_{order} = \begin{bmatrix} x_{(1),1} & x_{(1),2} & \cdots & x_{(1),p} \\ x_{(2),1} & x_{(2),2} & \cdots & x_{(2),p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{(n),1} & x_{(n),2} & \cdots & x_{(n),p} \end{bmatrix} \tag{1.7}$$

where $x_{(1),j} \leq x_{(2),j} \leq x_{(3),j} \leq \cdots \leq x_{(n),j}$ are the order statistics of the jth column of X. In each column, the minimum $x_{(1),j}$ is replaced by the second smallest value $x_{(2),j}$. Similarly, the maximum in each column $x_{(n),j}$ is replaced by the penultimate order statistic $x_{(n-1),j}$. The result is the ordered, truncated sample matrix.

$$X_{T,order} = \begin{bmatrix} x_{(2),1} & x_{(2),2} & \cdots & x_{(2),p} \\ x_{(2),1} & x_{(2),2} & \cdots & x_{(2),p} \\ x_{(3),1} & x_{(3),2} & \cdots & x_{(3),p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{(k),1} & x_{(k),2} & \cdots & x_{(k),p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{(n-2),1} & x_{(n-2),2} & \cdots & x_{(n-2),p} \\ x_{(n-1),1} & x_{(n-1),2} & \cdots & x_{(n-1),p} \\ x_{(n-1),1} & x_{(n-1),2} & \cdots & x_{(n-1),p} \end{bmatrix} \tag{1.8}$$

From this matrix, compute the truncated sample mean vector $\bar{\boldsymbol{x}}_T$ and the truncated sample standard deviation matrix Σ_T.

$$\left.\begin{aligned} \bar{\boldsymbol{x}}_T &= [\bar{x}_{T,1}, \bar{x}_{T,2}, \ldots, \bar{x}_{T,p}] \in \mathbb{R}^p \\ \bar{x}_{T,\ell} &= \frac{1}{n}\left(x_{(2),\ell} + x_{(n-1),\ell} + \sum_{j=2}^{n-1} x_{(j),\ell} \right),\ \ell = 1, 2, \ldots, p \end{aligned}\right\} \tag{1.9}$$

$$\left.\begin{aligned} \Sigma_T &= \begin{bmatrix} s_{T,1}^2 & 0 & \cdots & 0 \\ 0 & s_{T,2}^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & s_{T,p}^2 \end{bmatrix} \\ s_{T,\ell}^2 &= \frac{1}{n-1}\left((x_{(2),\ell} - \bar{x}_{T,\ell})^2 + (x_{(n-1),\ell} - \bar{x}_{T,\ell})^2 + \sum_{j=2}^{n-1} (x_{(j),\ell} - \bar{x}_{T,\ell})^2 \right) \end{aligned}\right\} \tag{1.10}$$

The metric used to determine the utility of a measurement is the *Mahalanobis distance* with respect to a *weighting matrix M*.

$$d_M^2(\boldsymbol{x}_k, \bar{\boldsymbol{x}}_T) = \left(\boldsymbol{x}_k - \bar{\boldsymbol{x}}_T\right) \cdot M^{-1} \cdot \left(\boldsymbol{x}_k - \bar{\boldsymbol{x}}_T\right)^T \tag{1.11}$$

If the number of samples n is sufficiently large (e.g., $n \geq 2 + {}^1\!/_2\ p(p+1)$), then the sample truncated covariance matrix $\text{cov}(X_T)$ of (1.12) can be used in place of the weighting matrix M in (1.11).

$$\text{cov}(X_T) = \frac{1}{n-3}(X_T - \mathbf{1}_{n\times 1} \cdot \bar{\boldsymbol{x}}_T)^T (X_T - \mathbf{1}_{n\times 1} \cdot \bar{\boldsymbol{x}}_T) \tag{1.12}$$

Otherwise, set $M = \Sigma_T$. Observe that the normalization factor in the covariance matrix formula (1.12) is $1/(n-3)$ rather than the usual $1/(n-1)$ since the first and last rows of X_T are duplicates of rows 2 and $n-1$. Thus, rather than n independent measurements, there are only $n-2$.

The Mahalanobis distance is distributed as a χ^2-random variable with p degrees of freedom regardless of choice of weighting matrix. If $\chi_p^2(\gamma)$ is the $\gamma\cdot 100\%$ quantile of a χ_p^2 distribution, then any measurement $\boldsymbol{x}_k$ whose Mahalanobis distance exceeds the value of $\chi_p^2(\gamma)$ is characterized as an outlier. That is, $d_M^2(\boldsymbol{x}_k, \bar{\boldsymbol{x}}_T) > \chi_p^2(\gamma)$ implies that $\boldsymbol{x}_k$ is an outlier. In parallel to the example illustrated in Figure 1.9, an *independent identically distributed* (*i.i.d.*) sample of 100 taken from $\mathcal{N}(0,1)$ along with two *i.i.d.* samples taken from $\mathcal{N}(0,100)$ are collected as the vector $\boldsymbol{x}$. Similarly, an additional *i.i.d.* sample of 100 taken from $\mathcal{N}(0,1)$ along with two *i.i.d.* samples taken from $\mathcal{N}(0,100)$ are collected as the vector $\boldsymbol{y}$. These data are examined for outliers using the

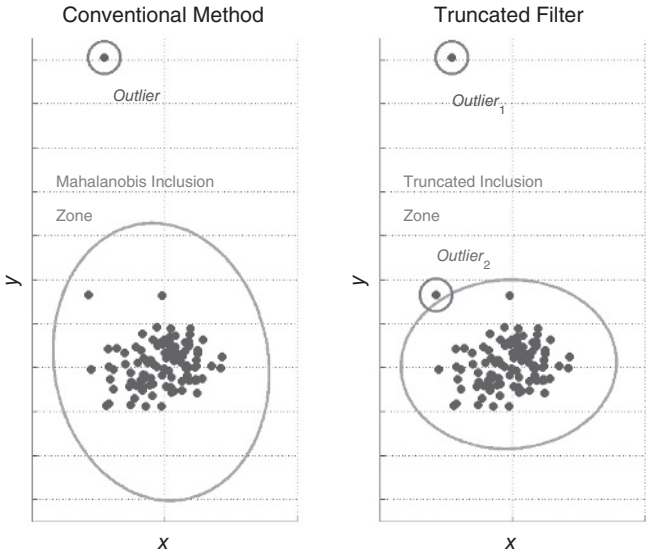

FIGURE 1.10 Two-dimensional outliers identified via conventional and truncated filter methods.

conventional and truncated filter methods. The conventional outlier method calculates the Mahalanobis distance (1.11) using the entire data matrix X to form either the covariance matrix (1.12) or diagonal matrix (1.10). The conventional method is only able to identify one of the two distinct measurements while the truncated filter method identifies both $\mathcal{N}(0, 100)$ samples as outliers. Moreover, the (Mahalanobis) inclusion zone $\mathcal{I}_M(\gamma) = \{\boldsymbol{x} \in \mathbb{R}^P | d_M^2(\boldsymbol{x}, \bar{\boldsymbol{x}}_T) \leq \chi_2^2(\gamma)\}$ is more compact for the truncated filter than it is for the conventional method. In this case, $\gamma = 0.999$. The results are presented in Figure 1.10.

MATLAB Commands
(*Two-Dimensional Outliers*)

```
% Data directory
Ddir = 'C:\Database\Math_Biology\Chapter_1\Data';
% Load the 2–dimensional outlier data
load(fullfile(Ddir,'X_2D_Outlier.mat'));
load(fullfile(Ddir,'Y_2D_Outlier.mat'));
% Compute the data covariance matrix M and mean vector m
A = [x,y]; M = cov(A); m = mean(A);
% χ^2–quantile
c = chi2inv(0.999,2);
% Mahalanobis distance
d = mahalanobis(A,m,M);
```

```
% Indices …
ii = (d.^2 > c);
% … and list of (conventional) outliers
A(ii,:) =
   -2.2761   14.1121
% Compute the truncated filter outliers
[Aout,iOut,S] = toutlier2(A,c);
A(iOut,:) =
   -2.8610    3.3224
   -2.2761   14.1121
```

EXERCISES

1.4 Use the data contained in the MAT-files `X_2D_Outlier.mat` and `Y_2D_Outlier.mat` along with the MATLAB commands contained in the table above to determine the outliers for the two-dimensional data set $[\boldsymbol{x}, \boldsymbol{y}]$ using different quantiles. Do the computations above for $\chi^2_2(\gamma)$ with $\gamma = 0.9$, 0.975, and 0.99. How does this choice of quantile affect the outlier selection for the conventional and truncated filter method? These data are obtained via the MATLAB commands

```
Ddir = 'C:\Database\Math_Biology\Chapter_1\Data';
load(fullfile(Ddir,'X_2D_Outlier.mat'));
load(fullfile(Ddir,'Y_2D_Outlier.mat'));
```

1.4 DATA CLUSTERING

There are collections of measurements from a particular class of objects that can be separated into two or more distinct subclasses. For example, the large (and imprecise) class of *automobiles* can be separated into sedans, station wagons, sport utility vehicles, and crossover vehicles. Each subclass, in turn, can be further refined via brand (e.g., Ford, VW, Toyota, etc.). This class distillation is particularly prominent in biology. Disease states, viral strains, and common food crops are often lumped into large and ill-defined categories. For the purposes of automated identification, more precise and well-defined subclasses can produce better classification.

Thus, this section will be concerned with the mathematical approach used to separate multivariate data of a single class into distinct subclasses.

Suppose a measurement $\boldsymbol{v}_0 \in \mathbb{R}^2$ is compared against two classes $Class_1$ and $Class_2$ as represented by the data matrices X and Y and illustrated in Figure 1.11. If the *Mahalanobis distance* (see equations (1.11) and (4.4) of Chapter 4) is the metric of proximity with weight matrix M equal to the class covariance, then the vector $\boldsymbol{v}_0$ is closer to $Class_1$ (dots •) than it is to $Class_2$ (diamonds ◆) since its Mahalanobis distance to $Class_1$ is smaller than the comparable distance to $Class_2$. An examination

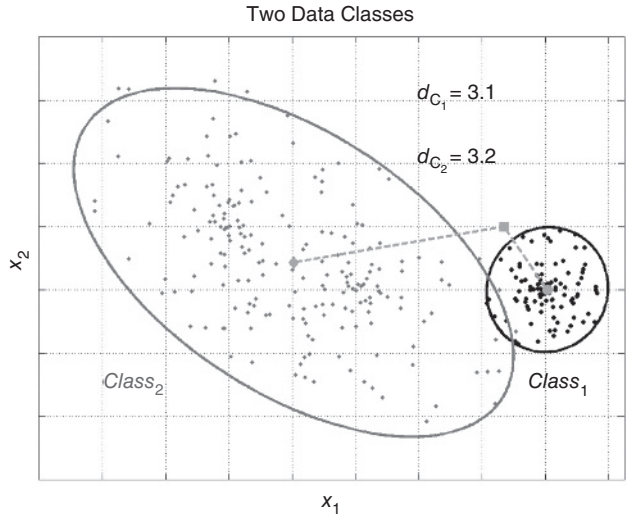

FIGURE 1.11 Mahalanobis distances for two distinct data classes.

of Figure 1.11, however, suggests that $Class_2$ has two concentrations or *clusters* of data within the error ellipse. If a mathematical process known as *data clustering* is applied to that data matrix Y for $Class_2$, it is seen that two distinct subclasses $Class_{2,1}$ (diamonds ◆) and $Class_{2,2}$ (stars ∗) arise. Moreover, these classes form a different partition of the classification space so that $\boldsymbol{v}_0$ has a smaller Mahalanobis distance to $Class_{2,1}$ than to either $Class_{2,2}$ or $Class_1$. Figure 1.12 demonstrates these remarks.

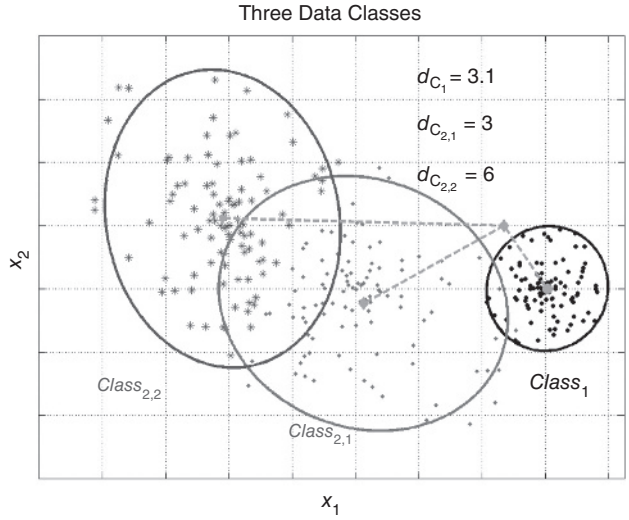

FIGURE 1.12 Mahalanobis distances for three distinct data classes.

What is this process of *data clustering* and how can it be applied to data matrices? The answer to this question is the focus of the remainder of this section.

1.4.1 Distances

The first step is to define the notion of a *distance*. Plainly the usual *Euclidean* distance from the selected point $\boldsymbol{v}_0$ appears closest to the $Class_1$ center. For a non-identity weighting matrix M, the Mahalanobis distance d_M can produce a different class assignment. Thus, the selection of a distance metric is critical to the classification process.

Definition 1.1 A *distance* d is a function mapping p-dimensional vectors onto the positive real line $\mathbb{R}^+$ so that

(i) $d : \mathbb{R}^p \otimes \mathbb{R}^p \to \mathbb{R}^+$
(ii) $d(\boldsymbol{x}, \boldsymbol{y}) \geq 0$ for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^p$ (*non-negativity*)
(iii) $d(\boldsymbol{x}, \boldsymbol{y}) = 0 \Leftrightarrow \boldsymbol{x} = \boldsymbol{y}$ (*reflexivity*)
(iv) $d(\boldsymbol{x}, \boldsymbol{y}) = d(\boldsymbol{y}, \boldsymbol{x})$ (*commutivity*)
(v) $d(\boldsymbol{x}, \boldsymbol{y}) \leq d(\boldsymbol{x}, z) + d(z, \boldsymbol{y})$ for any $\boldsymbol{x}, \boldsymbol{y}, z \in \mathbb{R}^p$. (*triangle inequality*)

The following are examples of popular distances.

Euclidean: $d(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\sum_{j=1}^{p} (x_j - y_j)^2}$

City Block (also called the *Manhattan* distance): $d(\boldsymbol{x}, \boldsymbol{y}) = \sum_{j=1}^{p} |x_j - y_j|$

Minkowski (or ℓ_q-norm): $d_q(\boldsymbol{x}, \boldsymbol{y}) = \left(\sum_{j=1}^{p} |x_j - y_j|^q \right)^{1/q}$

Mahalanobis: $d_M(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{(\boldsymbol{x} - \boldsymbol{y}) \cdot M^{-1} \cdot (\boldsymbol{x} - \boldsymbol{y})^T}$

Maximum (or Chebyshev): $d(\boldsymbol{x}, \boldsymbol{y}) = \max_{1 \leq j \leq p} |x_j - y_j|$

If two measurements have a "small" distance with respect to a data class, they can be thought of as *similar*. As with *distance* this notion can be formalized.

Definition 1.2 A *similarity measure* s is a metric that gauges the proximity of two measurements with respect to a distance d. Such a metric has the following properties.

(i) s: $\mathbb{R}^p \otimes \mathbb{R}^p \to [0, 1]$
(ii) $0 \leq s(\boldsymbol{x}, \boldsymbol{y}) \leq 1$ for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^p$
(iii) $s(\boldsymbol{x}, \boldsymbol{x}) = 1$ and $s(\boldsymbol{x}, \boldsymbol{y}) = s(\boldsymbol{y}, \boldsymbol{x})$

This formalism is a way to codify the idea that the smaller the distance in-between two measurements, the more "similar" they are. That is, $d(\boldsymbol{x}, \boldsymbol{y}) = 0$ should imply $s(\boldsymbol{x}, \boldsymbol{y}) = 1$. The connection in-between distance and similarity measure is summarized in the following results.

Theorem 1.1 If d is any distance metric, then $s(\boldsymbol{x}, \boldsymbol{y}) = \dfrac{|1 - d(\boldsymbol{x}, \boldsymbol{y})|}{1 + d^2(\boldsymbol{x}, \boldsymbol{y})}$ is a similarity measure.

Proof: Since d is a distance, then $s(\boldsymbol{x}, \boldsymbol{y}) = \dfrac{|1 - d(\boldsymbol{x}, \boldsymbol{y})|}{1 + d^2(\boldsymbol{x}, \boldsymbol{y})} = \dfrac{|1 - d(\boldsymbol{y}, \boldsymbol{x})|}{1 + d^2(\boldsymbol{y}, \boldsymbol{x})} = s(\boldsymbol{x}, \boldsymbol{y})$. Also, $s(\boldsymbol{x}, \boldsymbol{x}) = \dfrac{|1 - d(\boldsymbol{x}, \boldsymbol{x})|}{1 + d^2(\boldsymbol{x}, \boldsymbol{x})} = \dfrac{|1 + 0|}{1 + 0^2} = 1$. Thus, condition *iii* of Definition 1.2 is satisfied. Clearly, $s(\boldsymbol{x}, \boldsymbol{y}) \geq 0$ as it is defined as the ratio of a non-negative numerator and a positive denominator. Therefore, to show conditions *i* and *ii* of the definition, it remains to show that $s(\boldsymbol{x}, \boldsymbol{y}) \leq 1$ for any $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^p$. There are two cases to consider.

Case 1. $d(\boldsymbol{x}, \boldsymbol{y}) \geq 1$. In this case, $|1 - d(\boldsymbol{x}, \boldsymbol{y})| = d(\boldsymbol{x}, \boldsymbol{y}) - 1$ so that $0 \leq \dfrac{|1 - d(\boldsymbol{x}, \boldsymbol{y})|}{1 + d^2(\boldsymbol{x}, \boldsymbol{y})} = \dfrac{d(\boldsymbol{x}, \boldsymbol{y}) - 1}{1 + d^2(\boldsymbol{x}, \boldsymbol{y})} = \dfrac{1 - 1/d(\boldsymbol{x}, \boldsymbol{y})}{d(\boldsymbol{x}, \boldsymbol{y}) + 1/d(\boldsymbol{x}, \boldsymbol{y})}$. But this is the ratio of a number that is less than 1 (namely, $1 - 1/d(\boldsymbol{x}, \boldsymbol{y})$) and a number larger than 1 ($d(\boldsymbol{x}, \boldsymbol{y}) + 1/d(\boldsymbol{x}, \boldsymbol{y})$). Hence, the ratio is less than 1 and therefore, $s(\boldsymbol{x}, \boldsymbol{y}) \leq 1$.

Case 2. $d(\boldsymbol{x}, \boldsymbol{y}) < 1$. In this case, $0 \leq \dfrac{|1 - d(\boldsymbol{x}, \boldsymbol{y})|}{1 + d^2(\boldsymbol{x}, \boldsymbol{y})} = \dfrac{1 - d(\boldsymbol{x}, \boldsymbol{y})}{1 + d^2(\boldsymbol{x}, \boldsymbol{y})} < \dfrac{1}{1 + d^2(\boldsymbol{x}, \boldsymbol{y})} \leq 1$.

In either case, $0 \leq s(\boldsymbol{x}, \boldsymbol{y}) \leq 1$ for any $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^p$ so that conditions *i* and *ii* are satisfied. ■

Remarks:

1. If $d(\boldsymbol{x}, \boldsymbol{y}) < \varepsilon \ll 10^{-n}$ for "large" n, then $s(\boldsymbol{x}, \boldsymbol{y}) \approx \dfrac{1 - \varepsilon}{1 + \varepsilon^2} \approx 1$. Thus, small distances indicate a similarity near 1.
2. If $d(\boldsymbol{x}, \boldsymbol{y}) > N \gg 1$, then $s(\boldsymbol{x}, \boldsymbol{y}) \approx \dfrac{N}{1 + N^2} \approx \dfrac{1}{N} \to 0$ as $N \to +\infty$. Hence, large distances indicate a similarity near 0.

The next example combines the ideas of *distance* and *similarity metrics* to associate a measurement with a set of data classes.

Example 1.1: Suppose $\boldsymbol{v}_0 = [-0.65, 1]$ is a measurement, X is the data class matrix from $Class_1$, and Y is the data matrix from $Class_2$ of Figure 1.11. Table 1.1 summarizes the distances and similarity measures from $\boldsymbol{v}_0$ to each class with respect to

TABLE 1.1 Distance and similarity measures for Figure 1.11

Metric	$Class_1$	$Class_2$	Measure	Association
$d(\boldsymbol{v}, Class)$	0.34	0.64	Euclidean	1
$s(\boldsymbol{v}, Class)$	0.59	0.254		1
$d(\boldsymbol{v}, Class)$	3.12	3.20	Mahalanobis	1
$s(\boldsymbol{v}, Class)$	0.1975	0.1001		1
$d(\boldsymbol{v}, Class)$	0.298	0.51	Maximum	1
$s(\boldsymbol{v}, Class)$	0.6445	0.39		1
$d(\boldsymbol{v}, Class)$	0.4238	0.8992	City Block	1
$s(\boldsymbol{v}, Class)$	0.4885	0.0557		1

several of the distances presented at the beginning of this section. Notice that, in each case, the distance from $\boldsymbol{v}_0$ to the $Class_1$ data matrix X is smallest (simultaneously, the similarity of $\boldsymbol{v}_0$ to X is greatest) so that $\boldsymbol{v}_0$ would be associated with $Class_1$. Once a clustering method is applied to $Class_2$, however, the association will be reversed. This will be detailed in the next section.

1.4.2 The *K*-Means Method

Figures 1.11 and 1.12 illustrate the idea of taking a single data class and "splitting" it into two distinct subclasses. In the example mentioned earlier, the data from $Class_2$ were divided into the subclasses $Class_{2,1}$ and $Class_{2,2}$. How is this division achieved?

Suppose $X = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,p} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,p} \end{bmatrix}$ is the data matrix for a particular class of objects. If $\boldsymbol{x}_k = [x_{k,1}, x_{k,2}, \ldots, x_{k,p}]$ represents the kth row of X, then the data matrix can be viewed as a set of p-dimensional measurements: $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_p\}$. A *partition* of X is a collection of distinct subsets whose union is X. More precisely, $P = \{P_1, P_2, \ldots, P_K\}$ is a partition of X provided:

(i) $P_k \subset X$ for each $k = 1, 2, \ldots, K$

(ii) $X = \bigcup_{k=1}^{K} P_k$

(iii) $P_j \cap P_k = \varnothing$ for any $j \neq k$.

Let $\nu_k = |P_k|$ be the cardinality of the set P_k; that is, ν_k is the number of elements in the set P_k. The *attractor* $\boldsymbol{z}_k$ *for the kth member of a partition* P_k is the "average" of the elements within the partition element.

$$z_k = \frac{1}{\nu_k} \sum_{\boldsymbol{x} \in P_k} \boldsymbol{x} \tag{1.13}$$

The *K-means clustering method* requires the construction of a set of *K attractors* $Z = \{z_1, z_2, \ldots, z_K\} \subset X$ so that the associated partition $P = \{P_1, P_2, \ldots, P_K\}$ provides the maximum separation in-between the subclasses P_k with respect to a distance d. Therefore, the *K-means* algorithm is implemented to minimize the function

$$E(X) = \sum_{k=1}^{K} \sum_{\boldsymbol{x} \in P_k} d(\boldsymbol{x}, z_k) \tag{1.14}$$

with respect to a given partition P of X. Thus, if $\mathcal{P}_X$ is the collection of all partitions of X, the K-means method is to find the partition P so that $E(X)$ is minimized.

$$\min_{P \in \mathcal{P}_X} E(X) = \min_{P \in \mathcal{P}_X} \sum_{k=1}^{K} \sum_{\boldsymbol{x} \in P_k} d(\boldsymbol{x}, z_k) \tag{1.15}$$

Example 1.1 (*Continued*): After splitting $Class_2$ into two subclasses $Class_{2,1}$ and $Class_{2,2}$, calculate the distance of $\boldsymbol{v}_0 = [-0.65, 1]$ to these data classes along with the original $Class_1$. Now the choice of distance metric is crucial as $\boldsymbol{v}_0$ is closer to $Class_{2,1}$ than either $Class_{2,2}$ or $Class_1$ with respect to the Mahalanobis distance. Otherwise, $\boldsymbol{v}_0$ is associated with $Class_1$. Table 1.2 summarizes the computations.

The selection of distance metric is crucial to object classification. It also plays a crucial role in data clustering. This is the *art* of applied mathematics: The selection of methods and techniques that best respond to the data at hand. The general rule of thumb *for classification* is to use the *Mahalanobis distance provided the weight matrix M is well conditioned*. The Euclidean distance, whose weight matrix is the identity I, is recommended for the K-means clustering algorithm. For a well-conditioned weight matrix M, the Mahalanobis distance will separate classes along the eigenvectors of M. The Euclidean metric separates classes along the standard basis in which the data operates ($\mathbb{R}^p$). Care then should be taken in understanding

TABLE 1.2 Distance and similarity measures for Figure 1.12

Metric	$Class_1$	$Class_{2,1}$	$Class_{2,2}$	Measure	Association
$d(\boldsymbol{v}, Class)$	0.34	0.64	2.81	Euclidean	1
$s(\boldsymbol{v}, Class)$	0.59	0.254	0.204		1
$d(\boldsymbol{v}, Class)$	3.12	3.045	5.985	Mahalanobis	2
$s(\boldsymbol{v}, Class)$	0.1975	0.199	0.135		2
$d(\boldsymbol{v}, Class)$	0.298	0.5086	2.62	Maximum	1
$s(\boldsymbol{v}, Class)$	0.6445	0.39	0.206		1
$d(\boldsymbol{v}, Class)$	0.424	0.8992	3.4	City Block	1
$s(\boldsymbol{v}, Class)$	0.4885	0.0557	0.191		1

how the data are to be classified/clustered so that the end goal of providing the best categorization of the data is achieved. This requires some understanding of the underlying biological phenomenon being studied.

The example illustrated in Figures 1.11 and 1.12 is produced via the MATLAB commands shown below. In particular, `kmeans.m` (from the Statistics Toolbox) and `kcluster.m` are the MATLAB functions that perform the clustering via the *K*-means method.

MATLAB Commands
(*K-Means Clustering*)

```
% Data directory
Ddir = 'C:\PJC\Math_Biology\Chapter_1\Data';
% Retrieve the two-dimensional clustered data classes
load(fullfile(Ddir,'Clustered_Data_Classes1_2.mat'));
X = C{1}; Y = C{2};

% Split Class_2 into two subclasses
S = kcluster(Y,2);
% Identify the split classes as Z_1 and Z_2
Z1 = S.C{1}; Z2 = S.C{2};
```

1.4.3 Number of Clusters

As seen previously, the *K*-means clustering algorithm partitions a single data class into a specified number (K) of subclasses. What is the optimal number of subclasses? That is, is there a "best" K? This question is generally approached by way of *cluster validity indices*. These indices are a measure of how effective the cluster size is in reorganizing the data. There are *many* validity indices and the reader is referred to Gan et al. [7] for a thorough presentation of this topic. Instead, two indices with relatively straightforward interpretation are described and implemented on the data listed in the table "MATLAB Commands" from Section 1.4.2.

Before these indices are described, however, some notation must be established. The *data class matrix* $X \in \mathscr{Mat}_{n\times p}(\mathbb{R})$ will be treated as a collection of p-dimensional vectors $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}, \boldsymbol{x}_k \in \mathbb{R}^p$. The partition $P = \{P_1, P_2, \ldots, P_K\}$ of X with each $P_j \in \mathscr{Mat}_{n_j\times p}(\mathbb{R})$ and $\sum_{k=1}^{K} n_k = n$ is a *K-partition cluster* with *centroid element* $\boldsymbol{\mu}_k = \dfrac{1}{n_k}\sum_{\boldsymbol{x}\in P_k} \boldsymbol{x} \in \mathbb{R}^p$. Observe that $\boldsymbol{\mu}_k = [\mu_{k,1}, \mu_{k,2}, \ldots, \mu_{k,p}]$ implying the *mean of the centroid* is $\bar{\boldsymbol{\mu}}_k = \dfrac{1}{n_k}\sum_{\ell=1}^{n_k} \mu_{k,\ell}$. Finally, the *mean center of the data matrix* X is defined to be $\bar{X} = [\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_p]$ with $\bar{x}_k = \dfrac{1}{n}\sum_{j=1}^{n} x_{j,k} \equiv$ the column mean for $k = 1, 2, \ldots, p$.

1.4.3.1 The RMSSTD Validity Index The *root mean square standard deviation* (RMSSTD) index measures the variance of elements of a data class collection from its centroid elements. The optimal partition of the class collection occurs at the value j_0 at which the plot of the points $(j, RMS_{std}(j))$ attains a "knee," $j = 1, 2, \ldots, K$ (see Gan et al. [7, p. 310]. For any given partition $P = \{P_1, P_2, \ldots, P_K\}$ of X, let $\Pi_k = P_k - \mathbf{1}_{n_k \times 1} \cdot \mu_k$ where $\mathbf{1}_{n_k \times 1} = [1, 1, \ldots, 1]^T$ is the column vector of n_k ones. Then $\Pi_k \in \mathcal{Mat}_{n_k \times p}(\mathbb{R})$ for each k, $\pi_{i,j}^{(k)} = p_{i,k}^{(k)} - \mu_{k,j}$, and

$$\Pi_k = \begin{bmatrix} \pi_{1,1}^{(k)} & \pi_{1,2}^{(k)} & \cdots & \pi_{1,p}^{(k)} \\ \pi_{2,1}^{(k)} & \pi_{2,2}^{(k)} & \cdots & \pi_{2,p}^{(k)} \\ \vdots & \vdots & \ddots & \vdots \\ \pi_{n_k,1}^{(k)} & \pi_{n_k,2}^{(k)} & \cdots & \pi_{n_k,p}^{(k)} \end{bmatrix}$$

for each member of the partition.

$$RMS_{std} = \sqrt{\frac{\sum_{k=1}^{K} \sum_{i=1}^{n_k} \sum_{j=1}^{p} \left(\pi_{i,j}^{(k)}\right)^2}{p(n-K)}} \tag{1.16}$$

As is indicated in Figure 1.13, the "knee" of the validity curve for the RMS_{std} index occurs at $K = 2$.

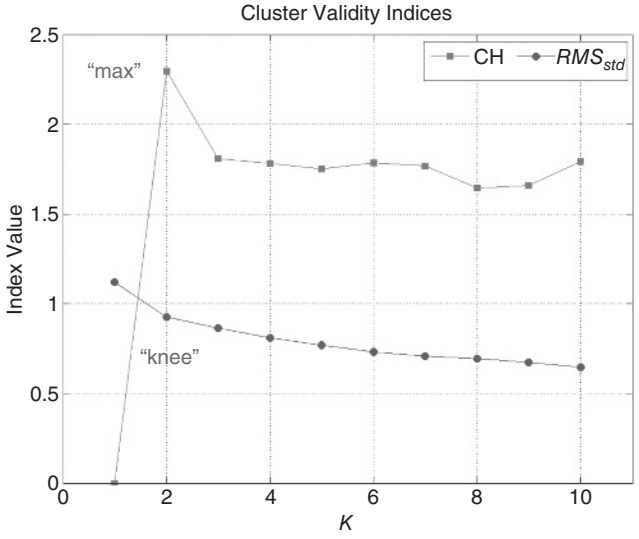

FIGURE 1.13 Validity curves for RMS_{std} and Calinski–Harabasz indices.

1.4.3.2 The Calinski–Harabasz Index This index is the ratio of the traces of the in-between-cluster scatter matrix and the within-cluster scatter matrix. The *Calinski–Harabasz* or *CH* index provides an optimal partition $P = \{P_1, P_2, \ldots, P_K\}$ of the data set X at the maximum value of the index.

$$CH = \frac{(n-K)\sum_{k=1}^{K} n_k(\boldsymbol{\mu}_k - \bar{X})(\boldsymbol{\mu}_k - \bar{X})^T}{(K-1)\sum_{k=1}^{K}\sum_{x \in P_k}(\boldsymbol{x} - \boldsymbol{\mu}_k)(\boldsymbol{x} - \boldsymbol{\mu}_k)^T} \tag{1.17}$$

An inspection of Figure 1.13 reveals that the maximum value of the *CH* index occurs at $K = 2$. Therefore, using either the *CH* or RMS_{std} indices, splitting $Class_2$ into two subclasses provides the optimal clustering effect.

EXERCISES

1.5 Try computing the *SD* and *SDbw* indices via the MATLAB functions `sdindex.m` and `sdbwindex.m` using the Mahalanobis distance options (see Gan et al. [7] for more details). Plot the results. What conclusions can be drawn from these indices? These data are obtained via the MATLAB commands

```
Ddir = 'C:\Data\Math_Biology\Chapter_1\Data';
load(fullfile(Ddir,'Clustered_Data_Classes1_2.mat'));
X = C{1}; Y = C{2};
```

1.6 Using the clustered data class contained in the MAT-file sited in the command table at the end of 1.4.2 (`Clustered_Data_Classes1_2.mat`), compute the distance and similarity metrics of the Minkowski measure using $q = 1$, 3, and 4. The commands listed in Exercise 1.5 will yield the desired data.

1.5 DATA QUALITY AND DATA CLEANING[2]

Data quality is the most significant and uncontrolled matter the data analyst/mathematical modeller must address. What is *data quality*? First, *data quality* refers to the state of the data under consideration. How were the data recorded? Were the data reviewed? What are the sources for data entry error? Can the data be edited and corrected for errant entries? These are *some* of the questions that the *quality control* specialist confronts. Hence, *data quality* should be thought of as a process in which the integrity of data can be ensured and safeguarded.

The issues that arise with respect to data quality vary considerably depending on the sources and kinds of measurements made. A significant portion of current interest

[2] The author wishes to thank Dr. William J. Satzer who inspired and co-wrote this section.

is focused on data mining, and thus often very large databases. There have been cases of passengers improperly flagged on "no-fly" lists due to spelling or transcription errors. If "John Doe" is a notorious terrorist while "Jon Dough" is an amiable pastry salesman, how can screening officials clearly identify friend from foe? And what to do with "J. Doe," "John B. Doe," and "John Deo?" With such a large list of names from which to flag an "undesirable" passenger, the accuracy of the data entry is crucial. Data quality problems can also arise, however, in other smaller scale settings. These need to be addressed in a systematic fashion.

Most students encounter only well managed or artificial data sets. Too often, textbooks or instructors provide "cleaned data" so that readers can apply various mathematical/statistical techniques to replicate examples. These data are close to perfect and do not represent the kinds of information that data modellers will actually encounter. No data are missing, no values are obvious typographical errors, and all the data are in one tidy file.

Indeed, one feature of this book is that, whenever possible, data sets are drawn from actual reports, studies, or genuine measurements.

The focus of the *quality control* specialist is to ensure data quality by *data cleaning*. This includes, but is not limited to, verifying data sources, reviewing data entry procedures, correcting misspellings, inversions, or inaccurate entries, missing information, invalid measurements, and identifying unlikely values. For example, if the heights and weights of dancers are presented as a data set, does the entry *Zoë Ballerina*, height $7'\ 2''$, weight 97 pounds make any sense?

Data cleaning aims to detect and remove errors and inconsistencies from data in order to improve its quality and prevent problems in subsequent data analyses. Sometimes data corruption occurs because of human error in data recording, and other times because of software or instrument error. When multiple data sources need to be integrated, the need for data cleaning is even more significant. In situations like this, data can be redundant, contradictory, and have multiple and inconsistent representations.

So far there is no science of data cleaning, only a collection of methods selected to meet needs of a particular environment. Dasu and Johnson [5], McCallum [10], Osborne [11], and Maydanchik [9] provide guidelines and insights into data cleaning. Eldén [6] examines the mathematics of *data mining*. A set of basic tenets for data extraction software is listed below. That is, when writing software that reads data from a set of measurements, the following recommendations are offered:

(i) Flag/identify any missing entry. If there is a vacant or missing data entry, replace the empty "value" with a NaN (an *IEEE* arithmetic representation for an object which is *not a number*).

(ii) Collect heading identifiers and indicate duplications. For example, if *Ht.*, *Height*, *H*, and *Hght* are column headers used to describe height measurements, note this naming disparity. Select a single column identifier and replace all "synonyms" with the representative.

(iii) Ensure consistent units. Again, if one column of data specified by *Height* provides measurements in *meters* while another column *H* lists measurements in *feet*, select a single unit and convert *feet* to *meters* (or vice versa).

(iv) Write code that verifies the data file being read and from which data are extracted. If there are two files, `DataFile1` and `DataFile2`, be certain that the extraction program is reading in the file that contains the requisite measurements. If reading data from Excel spreadsheets, check that the *sheet name* (i.e., the particular sheet/lower tab of the entire spreadsheet which contains data) is the one containing the desired information.

(v) Flag any column/heading that *should* be present in the data file but is not located via the extraction code.

(vi) Check for inconsistent information. If one column of data contains color information (e.g., red, green, or blue) and a second column contains RGB coordinates ([1, 0, 0] → red, [0, 1, 0] → green, and [0, 0, 1] → blue), check that the color (red) and coordinate ([1, 0, 0]) are consistent. If the (color, coordinate) pair is (green, [0, 0, 1]), then the measurement is *inconsistent*.

No exhaustive list of recommendations/guidelines can be given. The intention here is only to draw attention to the issue, identify some approaches, and offer select references to the developing literature in the area. The most important message of this section is the need to recognize that *data cleaning needs to precede data analysis*. No matter how insightful or complete is the model, the application of mathematics to poorly gathered data will result in nonsense or worse.

Data cleaning is variously known as "data scrubbing," "data wrangling," "data janitor work," as well as several other disparaging terms. It is regarded as a thankless job, yet interviews and expert estimates suggest that data cleaning can consume 50–80% of a data analyst's time. While considerable effort is being devoted to developing software that automates part of the task, much of it is essentially manual labor.

The aforementioned represent a few key ideas. They merit repetition:

(i) Inspect the data carefully before performing any analysis.

(ii) When possible, perform a visual inspection of the data files. To that end, this book provides a set of data visualization plotting functions (written in MATLAB).

(iii) Review the data (plotted or otherwise) to uncover anomalies such as missing data, mismatched data types, obvious outliers, data out-of-time sequence, and perhaps inconsistent scaling or mismatched measurement units within the data.

(iv) Document each data cleaning operation along the way.

This is the second crucial message of this section. *Document everything*: Merged files, obvious irregularities, dropped values, data transformations, and any new data derived from more raw forms. Further, always retain previous versions of data files;

never discard the original information. Otherwise, an error in data cleaning could effectively destroy the original data.

These recommendations listed are forged from many years of (often bitter) experience. They are offered in the hope that the reader will adopt a healthy skepticism toward data files and their processing. *Caveat emptor*.

REFERENCES

[1] Andrews, L. C. and B. K. Shivamoggi, *Integral Transforms for Engineers and Applied Mathematicians*, Macmillan Publishing Company, New York, 1988.

[2] Berra, Y., *The Yogi Book*, Workman Publishing, New York, 1998.

[3] Costa, P. J., *Bridging Mind and Model*, St. Thomas Technology Press, St. Paul, MN, 1994.

[4] Costa, P. J., "Truncated outlier filtering," *Journal of Biopharmaceutical Statistics*, vol. 24, pp. 1115–1129, 2014.

[5] Dasu, T. and T. Johnson, *Exploratory Data Mining and Data Cleaning*, John Wiley & Sons, Inc., Hoboken, NJ, 2003.

[6] Eldén, L., *Matrix Methods in Data Mining and Pattern Recognition*, SIAM Books, Philadelphia, PA, 2007.

[7] Gan, G., C. Ma, and J. Wu, *Data Clustering: Theory, Algorithms, and Applications*, ASA–SIAM Series on Statistics and Applied Probability, SIAM Books, Philadelphia, PA, ASA, Alexandria, VA, 2007.

[8] Johnson, R. A. and D. W. Wichern, *Applied Multivariate Statistical Analysis*, Fourth Edition, Prentice–Hall, Inc., Upper Saddle River, NJ, 1998.

[9] Maydanchik, A., *Data Quality Assessment*, Technics Publications, Bradley Beach, NJ, 2007.

[10] McCallum, Q. E., *The Bad Data Handbook, Cleaning Up The Data So You Can Get Back To Work*, O'Reilly Media, Sebastopol, CA, 2013.

[11] Osborne, J. W., *Best Practices in Data Cleaning: A Complete Guide to Everything You Need to do Before and After Collecting Your Data*, Sage Publications, Los Angeles, CA, 2013.

[12] Stewart, Potter, *United States Supreme Court Decision Jacobellis versus Ohio 1964*, 378 U.S. 184, https://en.wikipedia.org/wiki/Jacobellis_v._Ohio.

[13] Tholen, D.W., A. Kallner, J. W. Kennedy, J. S. Krouwer, and K. Meier, *Evaluation of Precision Performance of Quantitative Measurement Methods: Approved Guideline*, Second Edition, CLSI Document EP5–A2, vol. 24, no. 25, NCCLS, Wayne, PA, ISBN 1–56238–542–9.