Vectors and matrices are useful in representing multivariate data, and they occur naturally in working with linear equations or when expressing linear relationships among objects. Numerical algorithms for a variety of tasks involve matrix and vector arithmetic. An optimization algorithm to find the minimum of a function, for example, may use a vector of first derivatives and a matrix of second derivatives; and a method to solve a differential equation may use a matrix with a few diagonals for computing differences.

There are various precise ways of defining vectors and matrices, but we will generally think of them merely as linear or rectangular arrays of numbers, or scalars, on which an algebra is defined. Unless otherwise stated, we will assume the scalars are real numbers. We denote both the set of real numbers and the field of real numbers as \( \mathbb{R} \). (The field is the set together with the operators.) Occasionally we will take a geometrical perspective for vectors and will consider matrices to define geometrical transformations. In all contexts, however, the elements of vectors or matrices are real numbers (or, more generally, members of a field). When this is not the case, we will use more general phrases, such as “ordered lists” or “arrays”.

Many of the operations covered in the first few chapters, especially the transformations and factorizations in Chapter 5, are important because of their use in solving systems of linear equations, which will be discussed in Chapter 6; in computing eigenvectors, eigenvalues, and singular values, which will be discussed in Chapter 7; and in the applications in Chapter 9.

Throughout the first few chapters, we emphasize the facts that are important in statistical applications. We also occasionally refer to relevant computational issues, although computational details are addressed specifically in Part III.

It is very important to understand that the form of a mathematical expression and the way the expression should be evaluated in actual practice may be quite different. We remind the reader of this fact from time to time. That there is a difference in mathematical expressions and computational methods is one of the main messages of Chapters 10 and 11. (An example of this, in
notation that we will introduce later, is the expression $A^{-1}b$. If our goal is to solve a linear system $Ax = b$, we probably should never compute the matrix inverse $A^{-1}$ and then multiply it times $b$. Nevertheless, it may be entirely appropriate to write the expression $A^{-1}b$.)

1.1 Vectors

For a positive integer $n$, a vector (or $n$-vector) is an $n$-tuple, ordered (multi)set, or array of $n$ numbers, called *elements* or *scalars*. The number of elements is called the *order*, or sometimes the “length”, of the vector. An $n$-vector can be thought of as representing a point in $n$-dimensional space. In this setting, the “length” of the vector may also mean the Euclidean distance from the origin to the point represented by the vector; that is, the square root of the sum of the squares of the elements of the vector. This Euclidean distance will generally be what we mean when we refer to the *length* of a vector (see page 17).

We usually use a lowercase letter to represent a vector, and we use the same letter with a single subscript to represent an element of the vector.

The first element of an $n$-vector is the first ($1^{\text{st}}$) element and the last is the $n^{\text{th}}$ element. (This statement is not a tautology; in some computer systems, the first element of an object used to represent a vector is the $0^{\text{th}}$ element of the object. This sometimes makes it difficult to preserve the relationship between the computer entity and the object that is of interest.) We will use paradigms and notation that maintain the priority of the object of interest rather than the computer entity representing it.

We may write the $n$-vector $x$ as

$$ x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \quad \text{(1.1)} $$

or

$$ x = (x_1, \ldots, x_n). \quad \text{(1.2)} $$

We make no distinction between these two notations, although in some contexts we think of a vector as a “column”, so the first notation may be more natural. The simplicity of the second notation recommends it for common use. (And this notation does not require the additional symbol for transposition that some people use when they write the elements of a vector horizontally.)

We use the notation

$$ \mathbb{R}^n $$

to denote the set of $n$-vectors with real elements.
1.2 Arrays

Arrays are structured collections of elements corresponding in shape to lines, rectangles, or rectangular solids. The number of dimensions of an array is often called the rank of the array. Thus, a vector is an array of rank 1, and a matrix is an array of rank 2. A scalar, which can be thought of as a degenerate array, has rank 0. When referring to computer software objects, “rank” is generally used in this sense. (This term comes from its use in describing a tensor. A rank 0 tensor is a scalar, a rank 1 tensor is a vector, a rank 2 tensor is a square matrix, and so on. In our usage referring to arrays, we do not require that the dimensions be equal, however.) When we refer to “rank of an array”, we mean the number of dimensions. When we refer to “rank of a matrix”, we mean something different, as we discuss in Section 3.3. In linear algebra, this latter usage is far more common than the former.

1.3 Matrices

A matrix is a rectangular or two-dimensional array. We speak of the rows and columns of a matrix. The rows or columns can be considered to be vectors, and we often use this equivalence. An \( n \times m \) matrix is one with \( n \) rows and \( m \) columns. The number of rows and the number of columns determine the shape of the matrix. Note that the shape is the doubleton \( (n, m) \), not just a single number such as the ratio. If the number of rows is the same as the number of columns, the matrix is said to be square.

All matrices are two-dimensional in the sense of “dimension” used above. The word “dimension”, however, when applied to matrices, often means something different, namely the number of columns. (This usage of “dimension” is common both in geometry and in traditional statistical applications.)

We usually use an uppercase letter to represent a matrix. To represent an element of the matrix, we usually use the corresponding lowercase letter with a subscript to denote the row and a second subscript to represent the column. If a nontrivial expression is used to denote the row or the column, we separate the row and column subscripts with a comma.

Although vectors and matrices are fundamentally quite different types of objects, we can bring some unity to our discussion and notation by occasionally considering a vector to be a “column vector” and in some ways to be the same as an \( n \times 1 \) matrix. (This has nothing to do with the way we may write the elements of a vector. The notation in equation (1.2) is more convenient than that in equation (1.1) and so will generally be used in this book, but its use should not change the nature of the vector. Likewise, this has nothing to do with the way the elements of a vector or a matrix are stored in the computer.) When we use vectors and matrices in the same expression, however, we use the symbol “T” (for “transpose”) as a superscript to represent a vector that is being treated as a \( 1 \times n \) matrix.
We use the notation \( a_{*j} \) to correspond to the \( j \)th column of the matrix \( A \) and use \( a_{i*} \) to represent the (column) vector that corresponds to the \( i \)th row.

The first row is the 1st (first) row, and the first column is the 1st (first) column. (Again, we remark that computer entities used in some systems to represent matrices and to store elements of matrices as computer data sometimes index the elements beginning with 0. Furthermore, some systems use the first index to represent the column and the second index to indicate the row. We are not speaking here of the storage order—“row major” versus “column major”—we address that later, in Chapter 11. Rather, we are speaking of the mechanism of referring to the abstract entities. In image processing, for example, it is common practice to use the first index to represent the column and the second index to represent the row. In the software package PV-Wave, for example, there are two different kinds of two-dimensional objects: “arrays”, in which the indexing is done as in image processing, and “matrices”, in which the indexing is done as we have described.)

The \( n \times m \) matrix \( A \) can be written

\[
A = \begin{bmatrix}
a_{11} & \cdots & a_{1m} \\
\vdots & \ddots & \vdots \\
a_{n1} & \cdots & a_{nm}
\end{bmatrix}.
\] (1.3)

We also write the matrix \( A \) above as

\[
A = (a_{ij}),
\] (1.4)

with the indices \( i \) and \( j \) ranging over \( \{1, \ldots, n\} \) and \( \{1, \ldots, m\} \), respectively.

We use the notation \( A_{n\times m} \) to refer to the matrix \( A \) and simultaneously to indicate that it is \( n \times m \), and we use the notation

\[
\mathbb{R}^{n \times m}
\]
to refer to the set of all \( n \times m \) matrices with real elements.

We use the notation \( (A)_{ij} \) to refer to the element in the \( i \)th row and the \( j \)th column of the matrix \( A \); that is, in equation (1.3), \( (A)_{ij} = a_{ij} \).

Although vectors are column vectors and the notation in equations (1.1) and (1.2) represents the same entity, that would not be the same for matrices. If \( x_1, \ldots, x_n \) are scalars

\[
X = \begin{bmatrix}
x_1 \\
\vdots \\
x_n
\end{bmatrix}
\] (1.5)

and

\[
Y = [x_1, \ldots, x_n],
\] (1.6)

then \( X \) is an \( n \times 1 \) matrix and \( Y \) is a \( 1 \times n \) matrix (and \( Y \) is the transpose of \( X \)). Although an \( n \times 1 \) matrix is a different type of object from a vector,
we may treat $X$ in equation (1.5) or $Y^T$ in equation (1.6) as a vector when it is convenient to do so. Furthermore, although a $1 \times 1$ matrix, a 1-vector, and a scalar are all fundamentally different types of objects, we will treat a one by one matrix or a vector with only one element as a scalar whenever it is convenient.

One of the most important uses of matrices is as a transformation of a vector by vector/matrix multiplication. Such transformations are linear (a term that we define later). Although one can occasionally profitably distinguish matrices from linear transformations on vectors, for our present purposes there is no advantage in doing so. We will often treat matrices and linear transformations as equivalent.

Many of the properties of vectors and matrices we discuss hold for an infinite number of elements, but we will assume throughout this book that the number is finite.

**Subvectors and Submatrices**

We sometimes find it useful to work with only some of the elements of a vector or matrix. We refer to the respective arrays as “subvectors” or “submatrices”. We also allow the rearrangement of the elements by row or column permutations and still consider the resulting object as a subvector or submatrix. In Chapter 3, we will consider special forms of submatrices formed by “partitions” of given matrices.

**1.4 Representation of Data**

Before we can do any serious analysis of data, the data must be represented in some structure that is amenable to the operations of the analysis. In simple cases, the data are represented by a list of scalar values. The ordering in the list may be unimportant, and the analysis may just consist of computation of simple summary statistics. In other cases, the list represents a time series of observations, and the relationships of observations to each other as a function of their distance apart in the list are of interest. Often, the data can be represented meaningfully in two lists that are related to each other by the positions in the lists. The generalization of this representation is a two-dimensional array in which each column corresponds to a particular type of data.

A major consideration, of course, is the nature of the individual items of data. The observational data may be in various forms: quantitative measures, colors, text strings, and so on. Prior to most analyses of data, they must be represented as real numbers. In some cases, they can be represented easily as real numbers, although there may be restrictions on the mapping into the reals. (For example, do the data naturally assume only integral values, or could any real number be mapped back to a possible observation?)
The most common way of representing data is by using a two-dimensional array in which the rows correspond to observational units ("instances") and the columns correspond to particular types of observations ("variables" or "features"). If the data correspond to real numbers, this representation is the familiar $X$ data matrix. Much of this book is devoted to the matrix theory and computational methods for the analysis of data in this form. This type of matrix, perhaps with an adjoined vector, is the basic structure used in many familiar statistical methods, such as regression analysis, principal components analysis, analysis of variance, multidimensional scaling, and so on.

There are other types of structures that are useful in representing data based on graphs. A graph is a structure consisting of two components: a set of points, called vertices or nodes and a set of pairs of the points, called edges. (Note that this usage of the word "graph" is distinctly different from the more common one that refers to lines, curves, bars, and so on to represent data pictorially. The phrase "graph theory" is often used, or overused, to emphasize the present meaning of the word.) A graph $G = (V, E)$ with vertices $V = \{v_1, \ldots, v_n\}$ is distinguished primarily by the nature of the edge elements $(v_i, v_j)$ in $E$. Graphs are identified as complete graphs, directed graphs, trees, and so on, depending on $E$ and its relationship with $V$. A tree may be used for data that are naturally aggregated in a hierarchy, such as political unit, subunit, household, and individual. Trees are also useful for representing clustering of data at different levels of association. In this type of representation, the individual data elements are the leaves of the tree.

In another type of graphical representation that is often useful in "data mining", where we seek to uncover relationships among objects, the vertices are the objects, either observational units or features, and the edges indicate some commonality between vertices. For example, the vertices may be text documents, and an edge between two documents may indicate that a certain number of specific words or phrases occur in both documents. Despite the differences in the basic ways of representing data, in graphical modeling of data, many of the standard matrix operations used in more traditional data analysis are applied to matrices that arise naturally from the graph.

However, the data are represented, whether in an array or a network, the analysis of the data is often facilitated by using "association" matrices. The most familiar type of association matrix is perhaps a correlation matrix. We will encounter and use other types of association matrices in Chapter 8.