

2

Data and Databases

2.1 Introduction

Multivariate data consist of multiple measurements, observations, or responses obtained on a collection of selected variables. The types of variables usually encountered often depend upon those who collect the data (the *domain experts*), possibly together with some statistical colleagues; for it is these people who actively decide which variables are of interest in studying a particular phenomenon. In other circumstances, data are collected automatically and routinely without a research direction in mind, using software that records every observation or transaction made regardless of whether it may be important or not.

Data are raw facts, which can be numerical values (e.g., age, height, weight), text strings (e.g., a name), curves (e.g., a longitudinal record regarded as a single functional entity), or two-dimensional images (e.g., photograph, map). When data sets are “small” in size, we find it convenient to store them in *spreadsheets* or as *flat files* (large rectangular arrays). We can then use any statistical software package to import such data for subsequent data analysis, graphics, and inference. As mentioned in Chapter 1, massive data sets are now sprouting up everywhere. Data of such size need to be stored and manipulated in special database systems.

2.2 Examples

We first describe some examples of the data sets to be encountered in this book.

2.2.1 Example: DNA Microarray Data

The DNA (deoxyribonucleic acid) microarray has been described as “one of the great unintended consequences of the Human Genome Project” (Baker, 2003). The main impact of this enormous scientific achievement is to provide us with large and highly structured microarray data sets from which we can extract valuable genetic information. In particular, we would like to know whether “gene expression” (the process by which genetic information encoded in DNA is converted, first, into mRNA (messenger ribonucleic acid), and then into protein or any of several types of RNA) is any different for cancerous tissue as opposed to healthy tissue.

Microarray technology has enabled the *expression levels* of a huge number of genes within a specific cell culture or tissue to be monitored simultaneously and efficiently. This is important because differences in gene expression determine differences in protein abundance, which, in turn, determine different cell functions. Although protein abundance is difficult to determine, molecular biologists have discovered that gene expression can be measured indirectly through microarray experiments.

Popular types of microarray technologies include cDNA microarrays (developed at Stanford University) and high-density, synthetic, oligonucleotide microarrays (developed by Affymetrix, Inc., under the GENECHIP® trademark). Both technologies use the idea of hybridizing a “target” (which is usually either a single-stranded DNA or RNA sequence, extracted from biological tissue of interest) to a DNA “probe” (all or part of a single-stranded DNA sequence printed as “spots” onto a two-way grid of dimples in a glass or plastic microarray slide, where each spot corresponds to a specific gene).

The microarray slide is then exposed to a set of targets. Two biological mRNA samples, one obtained from cancerous tissue (the *experimental sample*), the other from healthy tissue (the *reference sample*), are reverse-transcribed into cDNA (complementary DNA); then, the reference cDNA is labeled with a green fluorescent dye (e.g., Cy3) and the experimental cDNA is labeled with a red fluorescent dye (e.g., Cy5). Fluorescence measurements are taken of each dye separately at each spot on the array. High gene expression in the tissue sample yields large quantities of hybridized cDNA, which means a high intensity value. Low intensity values derive from low gene expression.

The primary goal is to compare the intensity values, R and G, of the red and green channels, respectively, at each spot on the array. The most

popular statistic is the *intensity log-ratio*, $M = \log(R/G) = \log(R) - \log(G)$. Other such functions include the *probe value*, $PV = \log(R - G)$, and the *average log-intensity*, $A = \frac{1}{2}(\log R + \log G)$. The logarithm in each case is taken to base 2 because intensity values are usually integers ranging from 0 to $2^{16} - 1$.

Microarray data is a matrix whose rows are genes and whose columns are samples, although this row-column arrangement may be reversed. The genes play the role of variables, and the samples are the observations studied under different conditions. Such “conditions” include different experimental conditions (treatment vs. control samples), different tissue samples (healthy vs. cancerous tumors), and different time points (which may incorporate environmental changes).

For example, Figure 2.1 displays the heatmap for the expression levels of 92 genes obtained from a microarray study on 62 colon tissue samples, where the entries range from negative values (green) to positive values (red).¹ The tissue samples were derived from 40 different patients: 22 patients each provided both a normal tissue sample and a tumor tissue sample, whereas 18 patients each provided only a colon tumor sample. As a result, we have tumor samples from 40 patients ($T1, \dots, T40$) and normal samples from 22 patients (Normal1, \dots , Normal21), and this is the way the samples are labeled.

From the heatmap, we wish to identify expression patterns of interest in microarray data, focusing in on which genes contribute to those patterns across the various conditions. Multivariate statistical techniques applied to microarray data include supervised learning methods for classification and the unsupervised methods of cluster analysis.

2.2.2 Example: Mixtures of Polyaromatic Hydrocarbons

This example illustrates a very common problem in chemometrics. The data (Brereton, 2003, Section 5.1.2) come from a study of polyaromatic hydrocarbons (PAHs), which are described as follows:²

Polyaromatic hydrocarbons (PAHs) are ubiquitous environmental contaminants, which have been linked with tumors and effects on reproduction. PAHs are formed during the burning of coal, oil, gas, wood, tobacco, rubbish, and other organic

¹The data can be found in the file `alontop.txt` on the book’s website. The 92 genes are a subset of a larger set of more than 6500 genes whose expression levels were measured on these 62 tissue samples (Alon et al, 1999).

²This quote is taken from the August 1997 issue of the *Update* newsletter of the World Wildlife Fund–UK at its website www.wwf-uk.org/filelibrary/pdf/mu_32.pdf.

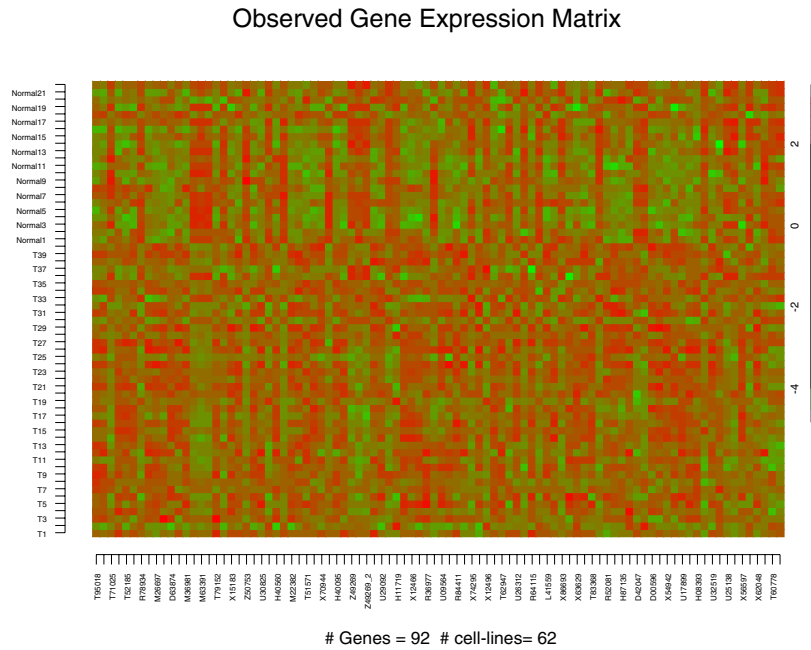


FIGURE 2.1. Gene expression heatmap of 92 genes (columns) and 62 tissue samples (rows) for the colon cancer data. The tissue samples are divided into 40 colon cancer samples (T1–T40) and 22 normal samples (Normal1–Normal22).

substances. They are also present in coal tars, crude oil, and petroleum products such as creosote and asphalt. There are some natural sources, such as forest fires and volcanoes, but PAHs mainly arise from combustion-related or oil-related man-made sources. A few PAHs are used by industry in medicines and to make dyes, plastics, and pesticides.

Table 2.1 gives a list of the 10 PAHs that are used in this example.

The data were collected in the following way.³ From the 10 PAHs listed in Table 2.1, 50 complex mixtures of certain concentrations (in mg L) of those PAHs were formed. From each such mixture, an electronic absorption

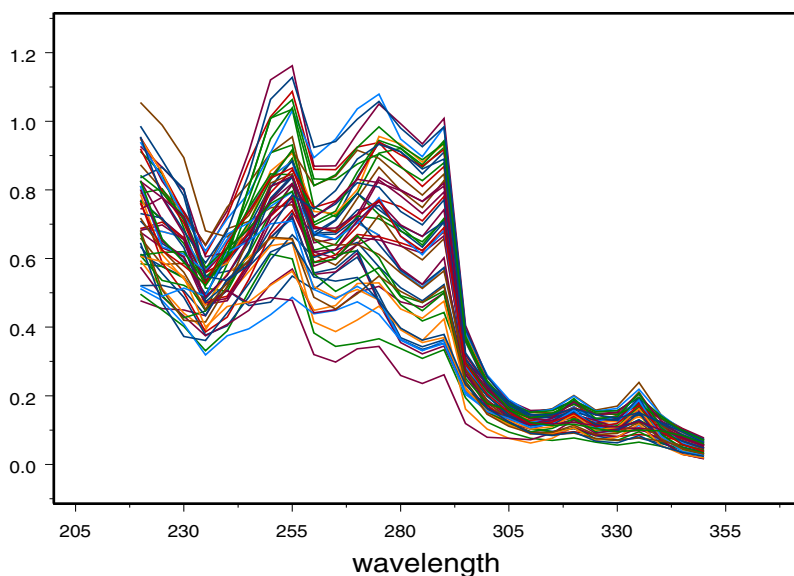
³The data, which can be found in the file `PAH.txt` on the book's website, can also be downloaded from the website `statmaster.sdu.dk/courses/ST02/data/index.html`. The fifty sample observations were originally divided into two independent sets, each of 25 observations, but were combined here so that we would have more observations than either set of data for the example.

TABLE 2.1. *Ten polyaromatic hydrocarbon (PAH) compounds.*

pyrene (Py), acenaphthene (Ace), anthracene (Anth), acenaphthylene (Acy), chrysene (Chry), benzantracene (Benz), fluoranthene (Fluora), fluorene (Fluore), naphthalene (Nap), phenanthracene (Phen)

spectrum (EAS) was computed. The spectra were then digitized at 5 nm intervals into $r = 27$ wavelength channels from 220 nm to 350 nm. The 50 spectra are displayed in Figure 2.2. The scatterplot matrix of the 10 PAHs is displayed in Figure 2.3. Notice that most of these scatterplots appear as 5×5 arrays of 50 points, where only half the points are visible because of a replication feature in the experimental design.

Using the resulting digitized values of the spectra, we wish to predict the individual concentrations of PAHs in the mixture. In chemometrics, this type of regression problem is referred to as *multivariate inverse calibration*: although the concentrations are actually the input variables and the spectrum values are the output variables in the chemical process, the real

**FIGURE 2.2.** *Electronic absorption spectroscopy (EAS) spectra of 50 samples of polyaromatic hydrocarbons (PAH), where the spectra are measured at 25 wavelengths within the range 220–350 nm.*

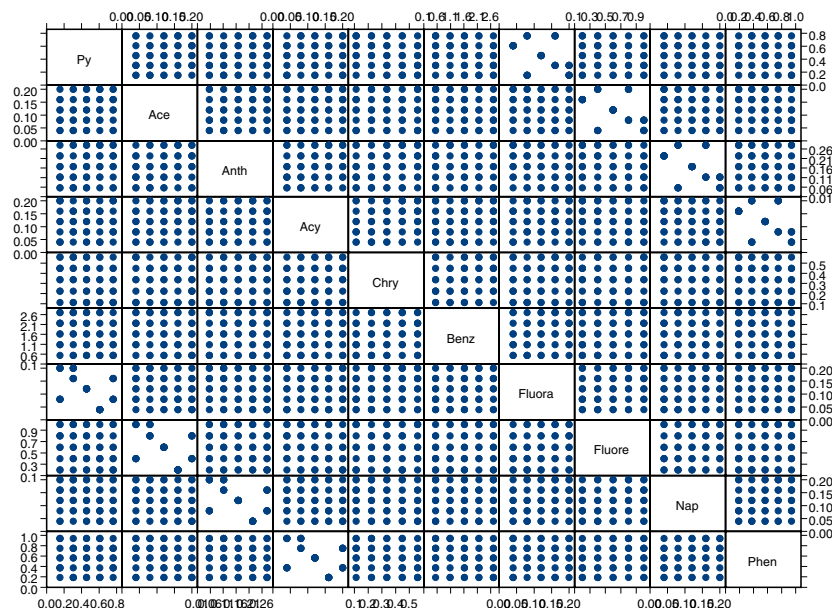


FIGURE 2.3. Scatterplot matrix of the mixture concentrations of the 10 chemicals in Table 2.1. In each scatterplot, there are 50 points; in most scatterplots, 25 of the points appear in a 5×5 array, and the other 25 are replications. In the remaining four scatterplots, there are eight distinguishable points with different numbers of replications.

goal is to predict the mixture concentrations (which are difficult to determine) from the spectra (easy to compute), and not vice versa.

2.2.3 Example: Face Recognition

Until recently, human face recognition was primarily based upon identifying individual facial features such as eyes, nose, mouth, ears, chin, head outline, glasses, and facial hair, and then putting them together computationally to construct a face. The most used approach today (and the one we describe here) is an innovative computerized system called *eigen-faces*, which operates directly on an image-based representation of faces (Turk and Pentland, 1991). Applications of such work include homeland security, video surveillance, human-computer interaction for entertainment purposes, robotics, and “smart” cards (e.g., passports, drivers’ licences, voter registration).

Each face, as a picture image, might be represented by a $(c \times d)$ -matrix of intensity values, which are usually quantized to 8-bit gray scale (0–255, with

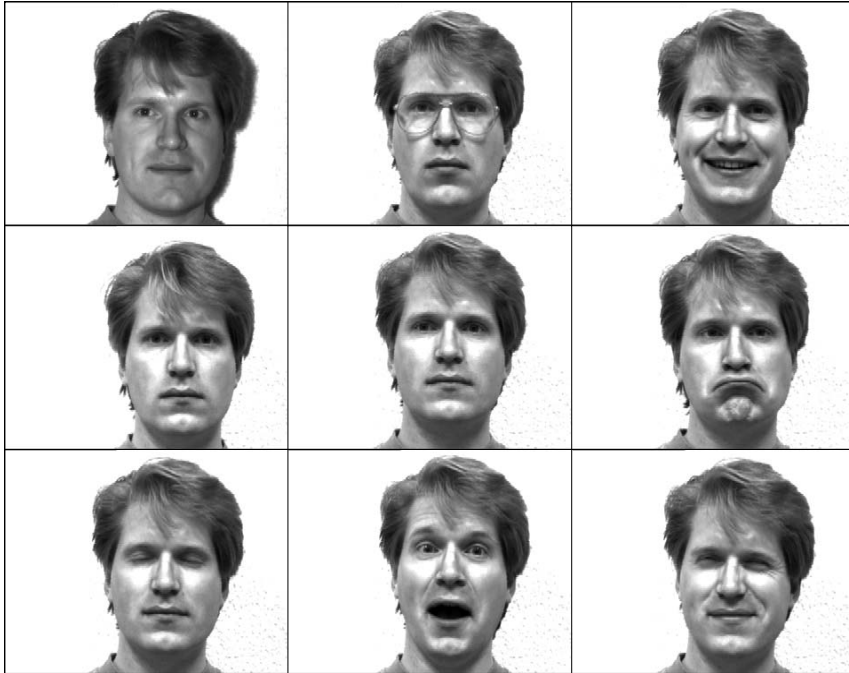


FIGURE 2.4. Face images of the same individual under nine different conditions (1=centerlight, 2=glasses, 3=happy, 4=no glasses, 5=normal, 6=sad, 7=sleepy, 8=surprised, 9=wink). From the Yale Face Database.

0 as black and 255 as white). These values are then scaled and converted to double precision, with values in $[0, 1]$. The values of c and d depend upon the degree of resolution needed. The matrix is then “vec’ed” by stacking the columns of the matrix under one another to form a cd -vector in *image space*. For example, if an image is digitized into a (256×256) -array of pixels, that face is now a point in a 65,536-dimensional space. We can view all possible images of one particular face as a lower-dimensional manifold (*face space*) embedded within the high-dimensional image space.

There are a number of repositories of face images. The data for this example were taken from the *Yale Face Database* (Belhumeur, Hespanha, and Kriegman, 1997).⁴ which contains 165 frontal-face grayscale images covering 15 individuals taken under 11 different conditions of different illumination (centerlight, leftlight, rightlight, normal), expression (happy, sad, sleepy, surprised, wink), and glasses (with and without). Each image has

⁴A list of the many face databases that can be accessed on the Internet, including the *Yale Face Database*, can be found at the website www.face-rec.org/databases.

size 320×243 , which then gets stacked into an r -vector, where $r = 77,760$. Figure 2.4 shows the images of a single individual taken under 9 of those 11 conditions. The problem is one of *dimensionality reduction*: what is the fewest number of variables necessary to identify these types of facial images?

2.3 Databases

A *database* is a collection of persistent data, where by “persistent” we mean data that can be removed from the database only by an explicit request and not through an application’s side effect. The most popular format for organizing data in a database is in the form of *tables* (also called *data arrays* or *data matrices*), each table having the form of a rectangular array arranged into rows and columns, where a row represents the values of all variables on a single multivariate *observation* (*response*, *case*, or *record*), and a column represents the values of a single *variable* for each observation.

In this book, a typical database table having n multivariate observations taken on r variables will be represented by an $(r \times n)$ -matrix,

$$\mathcal{X} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & & \vdots \\ x_{r1} & x_{r2} & \cdots & x_{rn} \end{pmatrix}, \quad (2.1)$$

say, having r rows and n columns. In (2.1), x_{ij} represents the value in the i th row ($i = 1, 2, \dots, r$) and j th column ($j = 1, 2, \dots, n$) of \mathcal{X} . Although database tables are set up to have the form of \mathcal{X}^T , with variables as columns and observations as rows, we will find it convenient in this book to set \mathcal{X} to be the transpose of the database table.

Databases exist for storing information. They are used for any of a number of different reasons, including statistical analysis, retrieving information from text-based documents (e.g., libraries, legislative records, case dockets in litigation proceedings), or obtaining administrative information (e.g., personnel, sales, financial, and customer records) needed for managing an organization. Databases can be of any size. Even small databases can be very useful if accessed often. Setting up a large and complex database typically involves a major financial commitment on the part of an organization, and so the database has to remain useful over a long time period. Thus, we should be able to extend a database as additional records become available and to correct, delete, and update records as necessary.

2.3.1 Data Types

Databases usually consist of mixtures of different types of variables:

Indexing: These are usually names, tags, case numbers, or serial numbers that identify a respondent or group of respondents. Their values may indicate the location where a particular measurement was taken, or the month or day of the year that an observation was made.

There are two special types of indexing variables:

1. A *primary key* is an indexing variable (or set of indexing variables) that uniquely identifies each observation in a database (e.g., patient numbers, account numbers).
2. A *foreign key* is an indexing variable in a database where that indexing variable is a primary key of a related database.

Binary: This is the simplest type of variable, having only two possible responses, such as YES or NO, SUCCESS or FAILURE, MALE or FEMALE, WHITE or NON-WHITE, FOR or AGAINST, SMOKER or NON-SMOKER, and so on. It is usually coded 0 or 1 for the two possible responses and is often referred to as a *dummy* or *indicator* variable.

Boolean: A *Boolean* variable has the two responses TRUE or FALSE but may also have the value UNKNOWN.

Nominal: This *character-string* data type is a more general version of a binary variable and has a fixed number of possible responses that cannot be usefully ordered. These responses are typically coded alphanumerically, and they usually represent disjoint classifications or categories set up by the investigator. Examples include the geographical location where data on other variables are collected, brand preference in a consumer survey, political party affiliation, and ethnic-racial identification of respondent.

Ordinal: The possible responses for this character-string data type are linearly ordered. An example is “excellent, good, fair, poor, bad, awful” (or “strongly disagree” to “strongly agree”). Another example is bond ratings for debt issues, recorded as AA+, AA, AA-, A+, A, A-, B+, B, and B-. Such responses may be assigned scores or rankings. They are often coded on a “ranking scale” of 1–5 (or 1–10). The main problem with these ranking scales is the implicit assumption of equidistance of the assigned scores. Brand preferences can sometimes be regarded as ordered.

Integer: The response is usually a nonnegative whole number and is often a count.

Continuous: This is a measured variable in which the continuity assumption depends upon a sufficient number of digits (and decimal places) being recorded. Continuous variables are specified as *numeric* or *decimal* in database systems, depending upon the precision required.

We note an important distinction between variables that are fixed and those that are stochastic:

Fixed: The values of a fixed variable have deliberately been set in advance, as in a designed experiment, or are considered “causal” to the phenomenon in question; as a result, interest centers only on a specific group of responses. This category usually refers to indexing variables but can also include some of the above types.

Stochastic: The values of a stochastic variable can be considered as having been chosen at random from a potential list (possibly, the real line or a portion of it) in some stochastic manner. In this sense, the values obtained are representative of the entire range of possible values of the variable in question.

We also need to distinguish between input and output variables:

Input variable: Also called a *predictor* or *independent variable*, typically denoted by X , and may be considered to be fixed (or preset or controlled) through a statistically designed experiment, or stochastic if it can take on values that are observed but not controlled.

Output variable: Also called a *response* or *dependent variable*, typically denoted by Y , and which is stochastic and dependent upon the input variables.

Most of the methods described in this book are designed to elicit information concerning the extent to which the outputs depend upon the inputs.

2.3.2 Trends in Data Storage

As data collections become larger and larger, and areas of research that were once “data-poor” now become “data-rich,” it is how we store those data that is of great importance.

For the individual researcher working with a relatively simple database, data are stored locally on hard disks. We know that hard-disk storage capacity is doubling annually (*Kryder’s Law*), and the trend toward tiny,

TABLE 2.2. *Internet websites containing many different databases.*

www.ics.uci.edu/pub/machine-learning-databases
lib.stat.cmu.edu/datasets
www.statsci.org/datasets.html
www.amstat.org/publications/jse/jse_data_archive.html
www.physionet.org/physiobank/database
biostat.mc.vanderbilt.edu/twiki/bin/view/Main/DataSets

high-capacity hard drives has outpaced even the rate of increase in number of transistors that can be placed on an integrated circuit (*Moore's Law*). Gordon E. Moore, Intel co-founder, predicted in 1965 that the number of transistors that can be placed on an integrated circuit would continue to increase at a constant rate for at least 10 years. In 1975, Moore predicted that the rate would double every two years. So far, this assessment has proved to be accurate, although Moore stated in 2005 that his law, which may hold for another two decades, cannot be sustained indefinitely.

Because chip speeds are doubling even faster than Moore had anticipated, we are seeing rapid progress toward the manufacturing of very small, high-performance storage devices. New types of data storage devices include three-dimensional *holographic storage*, where huge quantities (e.g., a terabyte) of data can be stored into a space the size of a sugar cube.

For large institutions, such as health maintenance organizations, educational establishments, national libraries, and industrial plants, data storage is a more complicated issue, and the primary storage facility is usually a remote “data warehouse.” We describe such storage facilities in Section 2.4.5.

2.3.3 Databases on the Internet

In Table 2.2, we list a few Internet websites from which databases of various sizes can be downloaded. Many of the data sets used as examples in this book were obtained through these websites.

There are also many databases available on the Internet that specialize in bioinformatics information, such as biological databases and published articles. These databases contain an amazingly rich variety of biological data, including DNA, RNA, and protein sequences, gene expression profiles, protein structures, transcription factors, and biochemical pathways. See Table 2.3 for examples of such websites.

A recent development in data-mining applications is the processing and categorization of natural-language text documents (e.g., news items, scientific publications, spam detection). With the rapid growth of the Internet and e-mail, academics, scientists, and librarians have shown enormous interest in mining the structured or unstructured knowledge present in large

collections of text documents. To help those whose research interests lie in analyzing text information, large databases (having more than 10,000 features) of text documents are now available.

For example, Table 2.4 lists a number of text databases. Two of the most popular collections of documents come from Reuters, Ltd., which is the world's largest text and television news agency; the English-language collections REUTERS-21578 containing 21,578 news items and RCV1 (*Reuters Corpus Volume 1*) (Lewis, Yang, Rose, and Li, 2004) containing 806,791 news items are drawn from online databases. The 20 Newsgroups database (donated by Tom Mitchell) contains 20,000 messages taken from 20 Usenet newsgroups. The OHSUMED text database (Hersh, Buckley, Leone, and Hickam, 1994) from Ohio State University contains 348,566 references and abstracts derived from Medline, an on-line medical information database, for the period 1987–1991.

Computerized databases of scientific articles (e.g., ARXIV, see Table 2.4) are assembled to (Shiffrin and Börner, 2004):

[I]dentify and organize research areas according to experts, institutions, grants, publications, journals, citations, text, and figures; discover interconnections among these; establish the import of research; reveal the export of research among fields; examine dynamic changes such as speed of growth and diversification; highlight economic factors in information production and dissemination; find and map scientific and social networks; and identify the impact of strategic and applied research funding by government and other agencies.

A common element of text databases is the dimensionality of the data, which can run well into the thousands. This makes visualization especially difficult. Furthermore, because text documents are typically noisy, possibly even having differing formats, some automated preprocessing may be necessary in order to arrive at high-quality, clean data. The availability of text databases in which preprocessing has already been undertaken is proving to be an important development in database research.

TABLE 2.3. *Internet websites containing microarray databases.*

www.broad.mit.edu/tools/data.html
sdmc.lit.org.sg/GEDatasets/Datasets.html
genome-www5.stanford.edu
www.bioconductor.org/packages/1.8/AnnotationData.html
www.ncbi.nlm.nih.gov/geo

TABLE 2.4. *Internet websites containing natural-language text databases.*

arXiv.org
medir.ohsu.edu/pub/ohsumed
kdd.ics.uci.edu/databases/reuters21578/reuters21578.html
kdd.ics.uci.edu/databases/20newsgroups/20newsgroups.html

2.4 Database Management

After data have been recorded and physically stored in a database, they need to be accessed by an authorized user who wishes to use the information. To access the database, the user has to interact with a database management system, which provides centralized control of all basic storage, access, and retrieval activities related to the database, while also minimizing duplications, redundancies, and inconsistencies in the database.

2.4.1 Elements of Database Systems

A *database management system* (DBMS) is a software system that manages data and provides controlled access to the database through a personal computer, an on-line workstation, or a terminal to a mainframe computer or network of computers. *Database systems* (consisting of databases, DBMS, and application programs) are typically used for managing large quantities of data. If we are working with a small data set with a simple structure, if the particular application is not complicated, and if multiple concurrent users (those who wish to access the same data at the same time) are not an issue, then there is no need to employ a DBMS.

A database system can be regarded as two entities: a *server* (or *backend*), which holds the DBMS, and a set of *clients* (or *frontend*), each of which consists of a hardware and a software component, including application programs that operate on the DBMS. Application programs typically include a query language processor, report writers, spreadsheets, natural language processors, and statistical software packages. If the server and clients communicate with each other from different machines through a *distributed processing network* (such as the Internet), we refer to the system as having a “client/server” architecture.

The major breakthrough in database systems was the introduction by 1970 of the *relational model*. We call a DBMS *relational* if the data are perceived by users only as *tables*, and if users can generate new tables from old ones. Tables in a *relational DBMS* (*RDBMS*) are rectangular arrays defined by their *rows* of observations (usually called *records* or *tuples*) and *columns* of variables (usually called *attributes* or *fields*); the number

of tuples is called the *cardinality*, and the number of attributes is called the *degree* of the table. A RDBMS contains operators that enable users to extract specified rows (**restrict**) or specified columns (**project**) from a table and match up (**join**) information stored in different tables by checking for common entries in common columns. Also part of a DBMS is a *data dictionary*, which is a system database that stores information (*metadata*) about the database itself.

2.4.2 Structured Query Language (SQL)

Users communicate with a RDBMS through a declarative *query language* (or general interactive enquiry facility), which is typically one of the many versions of SQL (*Structured Query Language*), usually pronounced “sequel” or “ess-cue-ell.” Created by IBM in the early 1970s and adopted as the industry standard in 1986, there are now many different implementations of SQL; no two are exactly the same, and each one is regarded as a *dialect*. In SQL, we can make a declarative statement that says, “From a given database, extract data that satisfy certain conditions,” and the DBMS has to determine how to do it.

SQL has two main sublanguages:

- a *data definition language* (DDL) is used primarily by database administrators to define data structures by creating a database object (such as a table) and altering or destroying a database object. It does not operate on data.
- a *data manipulation language* (DML) is an interactive system that allows users to retrieve, delete, and update existing data from and add new data to the database.

There is also a *data control language* (DCL), a security system used by the database administrator, which controls the privileges granted to database users.

Before creating a database consisting of multiple tables, it is advisable to do the following: give a unique name to each table; specify which columns each table should contain and identify their data types; to each table, assign a primary key that uniquely identifies each row of the table; and have at least one common column in each table in the database.

We can then build a working data set through the DDL by using SQL **create table** statements of the following form:

```
create table <table name> (<table elements>);
```

where <table name> specifies a name for the table and <table elements> is a list separated by commas that specifies column names, their data

types, and any column constraints. The set of data types depends upon the SQL dialect; they include: **char**(*c*) (a column of characters where *c* gives the maximum number of characters permitted in the column), **integer**, **decimal**(*a*, *b*) (where *a* is the total number of digits and *b* is the number of decimal places), **date** (in DBMS-approved format), and **logical** (True or False). The column constraints include **null** (that column may have empty row values) or **not null** (empty row values are not permitted in that column), primary keys, and any foreign keys. A semicolon ends the statement.

The DML includes such commands as **select** (allows users to retrieve specific database information), **insert** (adds new rows into an existing table), **update** (modifies information contained within a table), and **delete** (removes rows from a table). DML commands can be quite complicated and may include multiple expressions, clauses, predicates, or subqueries.

For example, the **select** statement (which supports **restrict**, **project**, and **join** operations, and is the most commonly used, but also most complicated SQL command) has the basic form

```
select <columns> from <table name> where <condition>;
```

where <columns> is a list of columns separated by commas. The **select** command is used to gather certain attributes from a particular RDBMS table, but where the tuples (rows) that are to be retrieved from those columns are limited to those that satisfy a given conditional Boolean search expression (i.e., True or False). One or more conditions may be joined by **and** or **or** operators as in set theory (the **and** always precedes the **or** operation). An asterisk may be used in place of the list of columns if all columns in the database are to be selected.

A primitive form of data analysis is included within the **select** statement through the use of five *aggregate operators*, **sum**, **avg**, **max**, **min**, and **count**, which provide the obvious *column statistics* over all rows that satisfy any stated conditions. For example, we can apply the command

```
select max(<column>) as max, min(<column>) as min from <table  
name> where <condition>;
```

to find the maximum (saved as “max”) and minimum (saved as “min”) of specified columns. Column statistics that are not aggregates (e.g., medians) are not available in SQL.

The smaller RDBMSs that are available include ACCESS (from Microsoft Corp.), MySQL (open source), and mSQL (Hughes Technologies). These “lightweight” RDBMSs can support a few hundred simultaneous users and up to a gigabyte of data. All of the major statistical software packages that operate in a Windows environment can import data stored in certain of these smaller RDBMSs, especially Microsoft ACCESS.

We note that purists strongly object to SQL being thought of as a relational query language because, they argue, it sacrifices many of the fundamental principles of the relational model in order to satisfy demands of practicality and performance. RDBMSs are slow in general and, because the dialects of SQL are different enough and are often incompatible with each other, changing RDBMSs can be a nightmarish experience. Even so, SQL remains the most popular RDBMS query language.

2.4.3 OLTP Databases

A large organization is likely to maintain a DBMS that manages a *domain-specific* database for the automatic capture and storage of real-time business transactions. This type of database is essential for handling an organization's day-to-day operations. An *on-line transaction processing* (OLTP) system is a DBMS application that is specially designed for very fast tracking of millions of small, simple transactions each day by a large number of concurrent users (tellers, cashiers, and clerks, who add, update, or delete a few records at a time in the database). Examples of OLTP databases include Internet-based travel reservations and airline seat bookings, automated teller machines (ATM) network transactions and point-of-sale terminals, transfers of electronic funds, stock trading records, credit card transactions and authorizations, and records of driving license holders.

These OLTP databases are dynamic in nature, changing almost continuously as transactions are automatically recorded by the system minute-by-minute. It is not unusual for an organization to employ several different OLTP systems to carry out its various business functions (e.g., point-of-sale, inventory control, customer invoicing). Although OLTP systems are optimized for processing huge numbers of short transactions, they are not configured for carrying out complex ad hoc and data analytic queries.

2.4.4 Integrating Distributed Databases

In certain situations, data may be distributed over many geographically dispersed sites (*nodes*) connected by a *communications network* (usually some sort of local-area network or wide-area network, depending upon distances involved). This is especially true for the healthcare industry. A huge amount of information, for example, on hospital management practices may be recorded from a number of different hospitals and consist of overlapping sets of variables and cases, all of which have to be combined (or integrated) into a single database for analysis.

Distributed databases also commonly occur in multicenter clinical trials in the pharmaceutical industry, where centers include institutions, hospitals, and clinics, sometimes located in several countries. The number of

total patients participating in such clinical trials rarely exceeds a few thousand, but there have been large-scale multicenter trials such as the Prostate Cancer Prevention Trial (Baker, 2001), which is a chemoprevention trial in which 18,000 men aged 55 years and older were randomized to either daily finasteride or placebo tablets for 7 years and involved 222 sites in the United States.

Data integration is the process of merging data that originate from multiple locations. When data are to be merged from different sources, several problems may arise:

- The data may be physically resident in computer files each of which was created using database software from different vendors.
- Different media formats may be used to store the information (e.g., audio or video tapes or DVDs, CDs or hard disks, hardcopy questionnaires, data downloaded over the Internet, medical images, scanned documents).
- The network of computer platforms that contain the data may be organized using different operating systems.
- The geographical locations of those platforms may be local or remote.
- Parts of the data may be duplicated when collected from different sources.
- Permission may need to be obtained from each source when dealing with sensitive data or security issues that will involve accessing personal, medical, business, or government records.

Faced with such potential inconsistencies, the information has to be integrated to become a consistent set of records for analysis.

2.4.5 Data Warehousing

An organization that needs to integrate multiple large OLTP databases will normally establish a single data warehouse for just that purpose. The term *data warehouse* was coined by W.H. Inmon to refer to a read-only, RDBMS running on a high-performance computer. The warehouse stores historical, detailed, and “scrubbed” data designed to be retrieved and queried efficiently and interactively by users through a dialect of SQL. Although data are not updated in realtime, fresh data can be added as supplements at regular intervals.

The components of a data warehouse are

DBMS: The publicly available RDBMSs that are almost mandatory for data warehousing usage include ORACLE (from Oracle Corp.), SQL SERVER (from Microsoft Corp.), SYBASE (from Sybase Inc.), POSTGRESQL (freeware), INFORMIX (from Informix Software, Inc.), and DB2 (from IBM Corp.). These “heavyweight” DBMSs can handle thousands of simultaneous users and can access up to several terabytes of data.

Hardware: It is generally accepted that large-scale data warehouse applications require either massively parallel-processing (MPP) or symmetric multiprocessing (SMP) supercomputers. Which type of hardware is installed depends upon many factors, including the complexity of the data and queries and the number of users that need to access the system.

- SMP architectures are often called “shared everything” because they share memory and resources to service more than a single CPU, they run a single copy of the operating system, and they share a single copy of each application. SMP is reputed to be better for those data warehouses whose capacity ranges between 50GB and 100GB.
- MPP architectures, on the other hand, are called “shared nothing”; they may have hundreds of CPUs in a single computer, each node of which is a self-contained computer with its own CPU, disk, and memory, and nodes are connected by a high-speed bus or switch. The larger the data warehouse (with capacity at least 200GB) and the more complex the queries, the more likely the organization will install an MPP server.

Such centralized data depositories typically contain huge quantities of information taking up hundreds of gigabytes or terabytes of disk space. Small data warehouses, which store subsets of the central warehouse for use by specialized groups or departments, are referred to as *data marts*.

More and more organizations that require a central data storage facility are setting up their own data warehouses and data marts. For example, according to Monk (2000), the Foreign Trade Division of the U.S. Census Bureau processes 5 million records each month from the U.S. Customs Service on 18,000 import commodities and 9,000 export commodities that travel between 250 countries and 50 regions within the United States. The raw import-export data are extracted, “scrubbed,” and loaded into a data warehouse having one terabyte of storage. Subsets of the data that focus on specific countries and commodities, together with two years of historical data, are then sent to a number of data marts for faster and more specific querying.

It has been reported that 90 percent of all Fortune 500 companies are currently (or soon will be) engaged in some form of data warehousing activity. Corporations such as Federal Express, UPS, JC Penney, Office Depot, 3M, Ace Hardware, and Sears, Roebuck and Co. have installed data warehouses that contain multi-terabytes of disk storage, and Wal-Mart and Kmart are already at the 100 terabyte range. These retailers use their data warehouses to access comprehensive sales records (extracted from the scanners of cash registers) and inventory records from thousands of stores worldwide.

Institutions of higher education now have data warehouses for information on their personnel, students, payroll, course enrollments and revenues, libraries, finance and purchasing, financial aid, alumni development, and campus data. Healthcare facilities have data warehouses for storing uniform billing data on hospital admissions and discharges, outpatient care, long-term care, individual patient records, physician licensing, certification, background, and specialties, operating and surgical profiles, financial data, CMS (Centers for Medicare and Medicaid Services) regulations, and nursing homes, and that might soon include image data.

2.4.6 *Decision Support Systems and OLAP*

The failure of OLTP systems to deliver analytical support (e.g., statistical querying and data analysis) of RDBMSs caused a major crisis in the database market until the concept of data warehouses each with its own *decision support system* (DSS) emerged. In a client/server computing environment, decision support is carried out using *on-line analytical processing* (OLAP) software tools.

There are two primary architectures for OLAP systems, ROLAP (*relational OLAP*) and MOLAP (*multidimensional OLAP*); in both, multivariate data are set up using a multidimensional model rather than the standard model, which emphasizes data-as-tables. The two systems store data differently, which in turn affects their performance characteristics and the amounts of data that can be handled.

ROLAP operates on data stored in a RDBMS. Complex multipass SQL commands can create various ad hoc multidimensional views of a two-dimensional data table (which slows down response times). ROLAP users can access all types of transactional data, which are stored in 100GB to multiple-terabyte data warehouses.

MOLAP operates on data stored in a specialized multidimensional DBMS. Variables are scaled categorically to allow transactional data to be pre-aggregated by all category combinations (which speeds up response times) and the results stored in the form of a “data cube” (a large, but sparse, multidimensional contingency table). MOLAP tools can handle up to 50GB of data stored in a data mart.

OLAP users typically access multivariate databases without being aware exactly which system has been implemented. There are other OLAP systems, including a hybrid version HOLAP.

The data analysis tools provided by a multidimensional OLAP system include operators that can *roll-up* (aggregate further, producing marginals), *drill-down* (de-aggregate to search for possible irregularities in the aggregates), *slice* (condition on a single variable), and *dice* (condition on a particular category) aggregated data in a multidimensional contingency table. Summary statistics that cannot be represented as aggregates (e.g., medians, modes) and graphics that need raw data for display (e.g., scatterplots, time series plots) are generally omitted from MOLAP menus (Wilkinson, 2005).

2.4.7 Statistical Packages and DBMSs

Some statistical analysis packages (e.g., SAS, SPSS) and MATLAB can run their complete libraries of statistical routines against their OLAP database servers.

A major effort is currently under way to provide a common interface for the S language (i.e., S-PLUS and particularly R) to access the really big DBMSs so that sophisticated data analysis can be carried out in a transparent manner (i.e., DBMS and platform independent). Although a *table* in a RDBMS is very similar to the concept of *data frame* in R and S-PLUS, there are many difficulties in building such interfaces.

The R package RODBC (written by Michael Lapsley and Brian Ripley, and available from CRAN) provides an R interface to DBMSs based upon the Microsoft ODBC (Open Database Connectivity) standard. RODBC, which runs on both MS Windows and Unix/Linux, is able to copy an R data frame to a table in a database (command: `sqlSave`), read a table from a DBMS into an R data frame (`sqlFetch`), submit an SQL query to an ODBC database (`sqlQuery`), retrieve the results (`sqlGetResults`), and update the table where the rows already exist (`sqlUpdate`). RODBC works with ORACLE, MS ACCESS, SYBASE, DB2, MYSQL, POSTGRESQL, and SQL SERVER on MS Windows platforms and with MYSQL, POSTGRESQL, and ORACLE under Unix/Linux.

2.5 Data Quality Problems

Errors exist in all kinds of databases. Those that are easy to detect will most likely be found at the data “cleaning” stage, whereas those errors that can be quite resistant to detection might only be discovered during data analysis. Data cleaning usually takes place as the data are received

and before they are stored in read-only format in a data warehouse. A consistent and cleaned-up version of the data can then be made available.

2.5.1 Data Inconsistencies

Errors in compiling and editing the resulting database are common and actually occur with alarming frequency, especially in cases where the data set is very large. When data from different sources are being connected, inconsistencies as to a person's name (especially in cases where a name can be spelled in several different ways) occur frequently, and matching (or "disambiguation") has to take place before such records can be merged. One popular solution is to employ Soundex (sound-indexing) techniques for name matching.

To get an idea of how poor data quality can become, consider the problem of estimating the extent of the undercount from census data collected for the 1990 U.S. census. Breiman (1994) identified a number of sources of error, including the following: *Matching errors* (incorrectly matching records from two different files of people with differing names, ages, missing gender or race identifiers, and different addresses), *fabrications* (the creation of fictitious people by dishonest interviewers), *census day address errors* (incorrectly recording the location of a person's residence on census day), *unreliable interviews* (many of the interviews were rejected as being unreliable), and *incomplete data* (a lack of specific information on certain members in the household). Most of the problems involving data fabrication, incomplete data, and unreliable interviews apparently occurred in areas that also had the highest estimated undercounts, such as the central cities and minority areas.

Massive data sets are prone to mistakes, errors, distortions, and, in general, poor data quality, just as is any data set, but such defects occur here on a far grander scale because of the size of the data set itself. When invalid product codes are entered for a product, they may easily be detected; when valid product codes, however, are entered for the wrong product, detection becomes more difficult. Customer codes may be entered inconsistently, especially those for gender identification (*M* and *F*, as opposed to 1 and 2). Duplication of records entered into the database from multiple sources can also be a problem. In these days of takeovers and buyouts, and mergers and acquisitions, what was once a code for a customer may now be a problem if the entity has since changed its description (e.g., Jenn-Air, Hoover, Norge, Magic Chef, etc., are all now part of Maytag Corp.). Any inconsistencies in historical data may also be difficult to correct if those who knew the answer are no longer with the company.

2.5.2 Outliers

Outliers are values in the data that, for one reason or another, do not appear to fit the pattern of the other data values; visually, they are located far away from the rest of the data. It is not unusual for outliers to be present in a data set.

Outliers can occur for many different reasons but should not be confused with *gross errors*. Gross errors are cases where “something went wrong” (Hampel, 2002); they include human errors (e.g., a numerical value recorded incorrectly) and mechanical errors (e.g., malfunctioning of a measuring instrument or a laboratory instrument during analysis). The density of gross errors depends upon the context and the quality of the data. In medical studies, gross error rates in excess of 10% have been quoted.

Univariate outliers are easy to detect when they indicate impossible (or “out of bounds”) values. More often, an outlier will be a value that is extreme, either too large or too small. For multivariate data, outlier detection is more difficult. Low-dimensional visual displays of the data (such as histograms, boxplots, scatterplots) can encourage insight into the data and provide at the same time a method for manually detecting some of the more obvious univariate or bivariate outliers.

When we have a large data set, outliers may not be all that rare. Unlike a data set of 100 or so observations, where we may find two or three outliers, in a data set of 100,000, we should not be surprised to discover a large number (in some cases, hundreds, and maybe even thousands) of outliers. For example, Figure 2.5 shows a scatterplot of the size (in bytes) of each of 50,000 packets⁵ containing roughly two minutes worth of TCP (transfer control protocol) packet traffic between Digital Equipment Corporation servers and the rest of the world on 8th March 1995 plotted against time. We see clear structure within the scatterplot: the vast majority of points occur within the 0–512 bytes range, and a number of dense horizontal bands occur inside this range; these bands show that the vast majority of packets sent consist of either 0 bytes (37% of the total packets), which are used only to acknowledge data sent by the other side, or 512 bytes (29% of the total packets). There are 952 packets each having more than 512 bytes, of which 137 points are identified as outliers (with values greater than 1.5 times IQR), including 61 points equal to the largest value, 1460 bytes.

To detect true multidimensional outliers, however, becomes a test of statistical ingenuity. A multivariate observation whose every component value may appear indistinguishable from the rest may yet be regarded as an outlier when all components are treated simultaneously. In large

⁵See www.amstat.org/publications/jse/datasets/packetdata.txt.

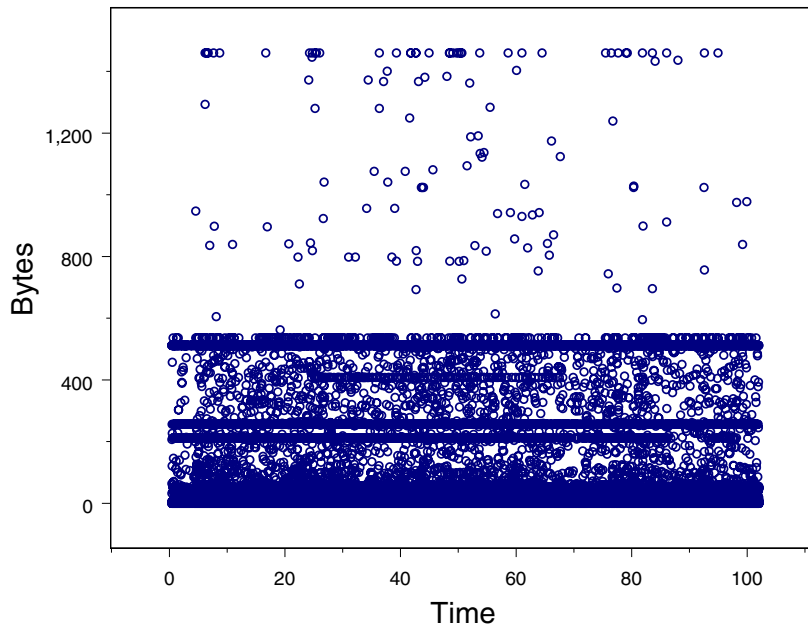


FIGURE 2.5. Time-series plot of 50,000 packets containing roughly two minutes worth of TCP (transfer control protocol) packets traffic between Digital Equipment Corporation servers and the rest of the world on 8th March 1995.

multivariate data sets, some combination of visual display of the data, manual outlier detection scheme, and automatic outlier detection program may be necessary: potential outliers could be “flagged” by an automatic screening device, and then an analyst would manually decide on the fate of that flagged outlier.

2.5.3 Missing Data

In the vast majority of data sets, there will be missing data values. For example, human subjects may refuse to answer certain items in a battery of questions because personal information is requested; some observations may be accidentally lost; some responses may be regarded as implausible and rejected; and in a study of financial records of a company, some records may not be available because of changes in reporting requirements and data from merged or reorganized organizations.

In R/S-PLUS, missing values are denoted by `NA`. In large databases, SQL incorporates the `null` as a *flag* or *mark* to indicate the absence of a data value, which might mean that the value is missing, unknown, nonexistent (no observation could be made for that entry), or that no value has yet

been assigned. A `null` is not equivalent to a zero value or to a text string filled with spaces. Sometimes, missing values are replaced by zeroes, other times by estimates of what they should be based on the rest of the data.

One popular method deletes those observations that contain missing data and analyzes only those cases that are observed in their entirety (often called *complete-case analysis* or *listwise-deletion method*). Such a complete-case analysis may be satisfactory if the proportion of deleted observations is small relative to the size of the entire data set and if the mechanism that leads to the missing data is independent of the variables in question — an assumption referred to by Donald Rubin as *missing at random* (MAR) or *missing completely at random* (MCAR) depending upon the exact nature of the *missing-data mechanism* (Little and Rubin, 1987). Any deleted observations may be used to help justify the MCAR assumption.

If the missing data constitute a sizeable proportion of the entire data set, then complete-case methods will not work. *Single imputation* has been used to impute (or “fill in”) an estimated value for each missing observation and then analyze the amended data set as if there had been no missing values in the first place. Such procedures include *hot-deck imputation*, where a missing value is imputed by substituting a value from a similar but complete record in the same data set; *mean imputation*, where the singly imputed value is just the mean of all the completely recorded values for that variable; and *regression imputation*, which uses the value predicted by a regression on the completely recorded data. Because sampling variability due to single imputation cannot be incorporated into the analysis as an additional source of variation, the standard errors of model estimates tend to be underestimated.

Since the late 1970s, Rubin and his colleagues have introduced a number of sophisticated algorithmic methods for dealing with incomplete data situations. One approach, the *EM algorithm* (Dempster, Laird, and Rubin, 1977; Little and Rubin, 1987), which alternates between an expectation (*E*) step and a maximization (*M*) step, is used to compute maximum-likelihood estimates of model parameters, where missing data are modeled as unobserved latent variables. We shall describe applications of the EM algorithm in more detail in later chapters of this book. A different approach, *multiple imputation* (Rubin, 1987), fills in the missing values $m > 1$ times, where the imputed values are generated each time from a distribution that may be different for each missing value; this creates m different data sets, which are analyzed separately, and then the m results are combined to estimate model parameters, standard errors, and confidence intervals.

2.5.4 More Variables than Observations

Many statistical computer packages do not allow the number of input variables, r , to exceed the number of observations, n , because, then, certain

matrices, such as the $(r \times r)$ covariance matrix, would have less than full rank, would be singular, and, hence, uninvertible. Yet, we should not be surprised when $r > n$. In fact, this situation occurs quite routinely in certain applications, and in such instances, r can be much greater than n . Typical examples include:

Satellite images When producing maps, remotely sensed image data are gathered from many sources, including satellite and aircraft scanners, where a few observations (usually fewer than 10 spectral bands) are measured at more than 100,000 wavelengths over a grid of pixels.

Chemometrics For determining concentrations in certain chemical compounds, calibration studies often need to analyze intensity measurements on a very large number (500–1,000 or more) of different spectral wavelengths using a small number of standard chemical samples.

Gene expression data Current microarray methods for studying human malignancies, such as tumors, simultaneously monitor expression levels of very large numbers of genes (5,000–10,000 or more) on relatively small numbers (fewer than 100) of tumor samples.

When $r > n$, one way of dealing with this problem is to analyze the data on each variable separately. However, this suggestion does not take account of correlations between the variables. Researchers have recently provided new statistical techniques that are not sensitive to the $r > n$ issue. We will address this situation in various sections of this book.

2.6 The Curse of Dimensionality

The term “curse of dimensionality” (Bellman, 1961) originally described how difficult it was to perform high-dimensional numerical integration. This led to the more general use of the term to describe the difficulty of dealing with statistical problems in high dimensions. Some implications include:

1. *We can never have enough data to cover every part of high-dimensional input space to learn which part of the space is important to a relationship and which is not.*

To see this, divide the axis of each of r input variables into K uniform intervals (or “bins”), so that the value of an input variable is approximated by the bin into which it falls. Such a partition divides the entire r -dimensional input space into K^r “hypercubes,” where K is chosen so that each hypercube contains at least one point in the input space. Given a specific hypercube in input space, an output value y_0 corresponding to a new input point in the hypercube can be approximated by computing some function

(e.g., the average value) of the y values that correspond to all the input points falling in that hypercube. Increasing K reduces the sizes of the hypercubes while increasing the precision of the approximation. However, at the same time, the number of hypercubes increases exponentially. If there has to be at least one input point in each hypercube, then the number of such points needed to cover all of r -space must also increase exponentially as r increases. In practice, we have a limited number of observations, with the result that the data are very sparsely spread around high-dimensional space.

2. *As the number of dimensions grows larger, almost all the volume inside a hypercubic region of input space lies closer to the boundary or surface of the hypercube rather than near the center.*

An r -dimensional hypercube $[-A, A]^r$ with each edge of length $2A$ has volume $(2A)^r$. Consider a slightly smaller hypercube with each edge of length $2(A - \epsilon)$, where $\epsilon > 0$ is small. The difference in volume between these two hypercubes is $(2A)^r - 2^r(A - \epsilon)^r$, and, hence, the proportion of the volume that is contained between the two hypercubes is

$$\frac{(2A)^r - 2^r(A - \epsilon)^r}{(2A)^r} = 1 - \left(1 - \frac{\epsilon}{A}\right)^r \rightarrow 1 \text{ as } r \rightarrow \infty.$$

In Figure 2.6, we see a graphical display of this result for $A = 1$ and number of dimensions $r = 1, 2, 10, 20, 50$. The same phenomenon also occurs with spherical regions in high-dimensional input space (see Exercise 2.4).

Bibliographical Notes

There are many different kinds of data sets and every application field measures items in its own way. The following issues of *Statistical Science* address the problems inherent with certain types of data: consumer transaction data and e-commerce data (May 2006), Internet data (August 2004), and microarray data (February 2003).

The Human Genome Project and Celera, a private company, simultaneously published draft accounts of the human genome in *Nature* and *Science* on 15th and 16th February 2001, respectively. An excellent article on gene expression is Sebastiani, Gussoni, Kohane, and Ramoni (2003). Books on the design and analysis of DNA microarray experiments and analyzing gene expression data are Drăghici (2003), Simon, Korn, McShane, Radmacher, Wright, and Zhao (2004), and the books edited by Parmigiani, Garrett, Irizarry, and Zeger (2003), Speed (2003), and Lander and Waterman (1995).

There are a huge number of books on database management systems. We found the books by Date (2000) and Connolly and Begg (2002) most useful. The concept of a “relational” database system originates with Codd (1970),

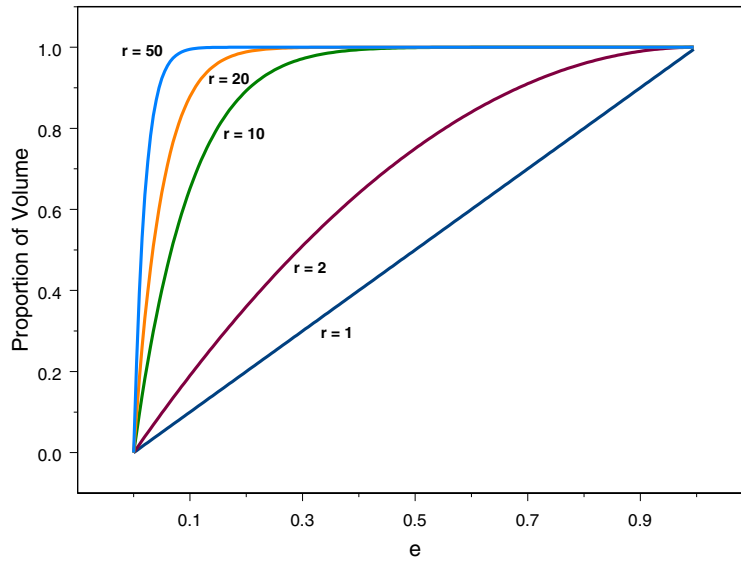


FIGURE 2.6. *Graphs of the proportion of the total volume contained between two hypercubes, one of edge length 2 and the other of edge length $2 - e$ for different numbers of dimensions r . As the number of dimensions increases, almost all the volume becomes closer to the surface of the hypercube.*

who received the 1981 ACM Turing Award for his work in the area. An excellent survey of the development and maintenance of biological databases and microarray repositories is given by Valdivia-Granda and Dwan (2006).

Books on missing data include Little and Rubin (1987) and Schafer (1997). A book on the EM algorithm is McLachlan and Krishnan (1997). For multiple imputation, see the book by Rubin (1987). Books on outlier detection include Rousseeuw and Leroy (1987) and Barnett and Lewis (1994).

Exercises

-
- 2.1** In a statistical application of your choice, what does a missing value mean? What are the traditional methods of imputing missing values in such an application?
- 2.2** In sample surveys, such as opinion polls, telephone surveys, and questionnaire surveys, nonresponse is a common occurrence. How would you design such a survey so as to minimize nonresponse?
- 2.3** Discuss the differences between single and multiple imputation for imputing missing data.

2.4 The volume of an r -dimensional sphere with radius A is given by $\text{vol}_r(A) = S_r A^r / r$, where $S_r = 2\pi^{r/2} / \Gamma(r/2)$ is the surface area of the unit sphere in r dimensions, $\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt = (x-1)!, 1x > 0$, is the gamma function, $\Gamma(x+1) = x\Gamma(x)$, and $\Gamma(1/2) = \pi^{1/2}$. Find the appropriate spherical volumes for two and three dimensions. Using a similar limiting argument as in (2) of Section 2.6, show that as the dimensionality increases, almost all the volume inside the sphere tends to be concentrated along a “thin shell” closer to the surface of the sphere than to the center.

2.5 Consider a hypercube of dimension r and sides of length $2A$ and inscribe in it an r -dimensional sphere of radius A . Find the proportion of the volume of the hypercube that is inside the hypersphere, and show that the proportion tends to 0 as the dimensionality r increases. In other words, show that all the density sits in the corners of the hypercube.

2.6 What are the advantages and disadvantages of database systems, and when would you find such a system useful for data analysis?

2.7 Find a commercial SQL product and discuss the various options that are available for the `create table` statement of that product.

2.8 Find a DBMS and investigate whether that system keeps track of database statistics. Which statistics does it maintain, how does it do that, and how does it update those statistics?

2.9 What are the advantages and disadvantages of distributed database systems?

2.10 (Fairley, Izenman, and Crunk, 2001) You are hired to carry out a survey of damage to the bricks of the walls of a residential complex consisting of five buildings, each having 5, 6, or 7 stories. The type of damage of interest is called spalling and refers to deterioration of the surface of the brick, usually caused by freeze-thaw weather conditions. Spalling appears to be high at the top stories and low at the ground. The walls consist of three-quarter million bricks. You take a photographic survey of all the walls of the complex and count the number of bricks in the photographs that are spalled. However, the photographs show that some portions of the walls are obscured by bushes, trees, pipes, vehicles, etc. So, the photographs are not a complete record of brick damage in the complex. Discuss how would you estimate the spall rate (spalls per 1,000 bricks) for the entire complex. What would you do about the missing data in your estimation procedure?

2.11 Read about MAR (missing at random) and MCAR (missing completely at random) and discuss their differences and implications for imputing missing data.



<http://www.springer.com/978-0-387-78188-4>

Modern Multivariate Statistical Techniques
Regression, Classification, and Manifold Learning

Izenman, A.J.

2008, XXV, 733 p., Hardcover

ISBN: 978-0-387-78188-4