

Part I

Introduction to Exploratory Data Analysis

Chapter 1

Introduction to Exploratory Data Analysis

*We shall not cease from exploration
And the end of all our exploring
Will be to arrive where we started
And know the place for the first time.*

T. S. Eliot, “Little Gidding” (the last of his *Four Quartets*)

The purpose of this chapter is to provide some introductory and background information. First, we cover the philosophy of exploratory data analysis and discuss how this fits in with other data analysis techniques and objectives. This is followed by an overview of the text, which includes the software that will be used and the background necessary to understand the methods. We then present several data sets that will be employed throughout the book to illustrate the concepts and ideas. Finally, we conclude the chapter with some information on data transforms, which will be important in some of the methods presented in the text.

1.1 What is Exploratory Data Analysis

John W. Tukey [1977] was one of the first statisticians to provide a detailed description of *exploratory data analysis* (EDA). He defined it as “detective work - numerical detective work - or counting detective work - or graphical detective work.” [Tukey, 1977, page 1] It is mostly a philosophy of data analysis where the researcher examines the data without any pre-conceived ideas in order to discover what the data can tell him about the phenomena being studied. Tukey contrasts this with *confirmatory data analysis* (CDA), an area of data analysis that is mostly concerned with statistical hypothesis testing, confidence intervals, estimation, etc. Tukey [1977] states that “Confirmatory data analysis is judicial or quasi-judicial in character.” CDA methods typically involve the process of making inferences about or estimates of some population characteristic and then trying to evaluate the

precision associated with the results. EDA and CDA should not be used separately from each other, but rather they should be used in a complementary way. The analyst explores the data looking for patterns and structure that leads to hypotheses and models.

Tukey's book on EDA was written at a time when computers were not widely available and the data sets tended to be somewhat small, especially by today's standards. So, Tukey developed methods that could be accomplished using pencil and paper, such as the familiar box-and-whisker plots (also known as boxplots) and the stem-and-leaf. He also included discussions of data transformation, smoothing, slicing, and others. Since this book is written at a time when computers are widely available, we go beyond what Tukey used in EDA and present computationally intensive methods for pattern discovery and statistical visualization. However, our philosophy of EDA is the same - that those engaged in it are *data detectives*.

Tukey [1980], expanding on his ideas of how exploratory and confirmatory data analysis fit together, presents a typical straight-line methodology for CDA; its steps follow:

1. State the question(s) to be investigated.
2. Design an experiment to address the questions.
3. Collect data according to the designed experiment.
4. Perform a statistical analysis of the data.
5. Produce an answer.

This procedure is the heart of the usual confirmatory process. To incorporate EDA, Tukey revises the first two steps as follows:

1. Start with some idea.
2. Iterate between asking a question and creating a design.

Forming the question involves issues such as: What can or should be asked? What designs are possible? How likely is it that a design will give a useful answer? The ideas and methods of EDA play a role in this process. In conclusion, Tukey states that EDA is an attitude, a flexibility, and some graph paper.

A small, easily read book on EDA written from a social science perspective is the one by Hartwig and Dearing [1979]. They describe the CDA mode as one that answers questions such as "Do the data confirm hypothesis XYZ?" Whereas, EDA tends to ask "What can the data tell me about relationship XYZ?" Hartwig and Dearing specify two principles for EDA: *skepticism* and *openness*. This might involve visualization of the data to look for anomalies or patterns, the use of resistant statistics to summarize the data, openness to the transformation of the data to gain better insights, and the generation of models.

Some of the ideas of EDA and their importance to teaching statistics were discussed by Chatfield [1985]. He called the topic *initial data analysis* or IDA. While Chatfield agrees with the EDA emphasis on starting with the noninferential approach in data analysis, he also stresses the need for looking at how the data were collected, what are the objectives of the analysis, and the use of EDA/IDA as part of an integrated approach to statistical inference.

Hoaglin [1982] provides a summary of EDA in the *Encyclopedia of Statistical Sciences*. He describes EDA as the “flexible searching for clues and evidence” and confirmatory data analysis as “evaluating the available evidence.” In his summary, he states that EDA encompasses four themes: resistance, residuals, re-expression and display.

Resistant data analysis pertains to those methods where an arbitrary change in a data point or small subset of the data yields a small change in the result. A related idea is **robustness**, which has to do with how sensitive an analysis is to departures from the assumptions of an underlying probabilistic model.

Residuals are what we have left over after a summary or fitted model has been subtracted out. We can write this as

$$\text{residual} = \text{data} - \text{fit}.$$

The idea of examining residuals is common practice today. Residuals should be looked at carefully for lack of fit, heteroscedasticity (nonconstant variance), nonadditivity, and other interesting characteristics of the data.

Re-expression has to do with the transformation of the data to some other scale that might make the variance constant, might yield symmetric residuals, could linearize the data or add some other effect. The goal of re-expression for EDA is to facilitate the search for structure, patterns, or other information.

Finally, we have the importance of **displays** or **visualization** techniques for EDA. As we described previously, the displays used most often by early practitioners of EDA included the stem-and-leaf plots and boxplots. The use of scientific and statistical visualization is fundamental to EDA, because often the only way to discover patterns, structure or to generate hypotheses is by visual transformations of the data.

Given the increased capabilities of computing and data storage, where massive amounts of data are collected and stored simply because we can do so and not because of some designed experiment, questions are often generated *after* the data have been collected [Hand, Mannila and Smyth, 2001; Wegman, 1988]. Perhaps there is an evolution of the concept of EDA in the making and the need for a new philosophy of data analysis.

1.2 Overview of the Text

This book is divided into two main sections: pattern discovery and graphical EDA. We first cover linear and nonlinear dimensionality reduction because sometimes structure is discovered or can only be discovered with fewer dimensions or features. We include some classical techniques such as principal component analysis, factor analysis, and multidimensional scaling, as well as some of the more recent computationally intensive methods like self-organizing maps, locally linear embedding, isometric feature mapping, and generative topographic maps.

Searching the data for insights and information is fundamental to EDA. So, we describe several methods that ‘tour’ the data looking for interesting structure (holes, outliers, clusters, etc.). These are variants of the grand tour and projection pursuit that try to look at the data set in many 2-D or 3-D views in the hope of discovering something interesting and informative.

Clustering or unsupervised learning is a standard tool in EDA and data mining. These methods look for groups or clusters, and some of the issues that must be addressed involve determining the number of clusters and the validity or strength of the clusters. Here we cover some of the classical methods such as hierarchical clustering and k -means. We also devote an entire chapter to a newer technique called model-based clustering that includes a way to determine the number of clusters and to assess the resulting clusters.

Evaluating the relationship between variables is an important subject in data analysis. We do not cover the standard regression methodology; it is assumed that the reader already understands that subject. Instead, we include a chapter on scatterplot smoothing techniques such as loess.

The second section of the book discusses many of the standard techniques of visualization for EDA. The reader will note, however, that graphical techniques, by necessity, are used throughout the book to illustrate ideas and concepts.

In this section, we provide some classic, as well as some novel ways of visualizing the results of the cluster process, such as dendrograms, treemaps, rectangle plots, and ReClus. These visualization techniques can be used to assess the output from the various clustering algorithms that were covered in the first section of the book. Distribution shapes can tell us important things about the underlying phenomena that produced the data. We will look at ways to determine the shape of the distribution by using boxplots, bagplots, q - q plots, histograms, and others.

Finally, we present ways to visualize multivariate data. These include parallel coordinate plots, scatterplot matrices, glyph plots, coplots, dot charts, and Andrews’ curves. The ability to interact with the plot to uncover structure or patterns is important, and we present some of the standard

methods such as linking and brushing. We also connect both sections by revisiting the idea of the grand tour and show how that can be implemented with Andrews' curves and parallel coordinate plots.

We realize that other topics can be considered part of EDA, such as descriptive statistics, outlier detection, robust data analysis, probability density estimation, and residual analysis. However, these topics are beyond the scope of this book. Descriptive statistics are covered in introductory statistics texts, and since we assume that readers are familiar with this subject matter, there is no need to provide explanations here. Similarly, we do not emphasize residual analysis as a stand-alone subject, mostly because this is widely discussed in other books on regression and multivariate analysis.

We do cover some density estimation, such as model-based clustering ([Chapter 6](#)) and histograms ([Chapter 9](#)). The reader is referred to Scott [1992] for an excellent treatment of the theory and methods of multivariate density estimation in general or Silverman [1986] for kernel density estimation. For more information on MATLAB implementations of density estimation the reader can refer to Martinez and Martinez [2002]. Finally, we will likely encounter outlier detection as we go along in the text, but this topic, along with robust statistics, will not be covered as a stand-alone subject. There are several books on outlier detection and robust statistics. These include Hoaglin, Mosteller and Tukey [1983], Huber [1981], and Rousseeuw and Leroy [1987]. A rather dated paper on the topic is Hogg [1974].

We use MATLAB® throughout the book to illustrate the ideas and to show how they can be implemented in software. Much of the code used in the examples and to create the figures is freely available, either as part of the downloadable toolbox included with the book or on other internet sites. This information will be discussed in more detail in Appendix B. For MATLAB product information, please contact:

The MathWorks, Inc.
3 Apple Hill Drive
Natick, MA, 01760-2098 USA
Tel: 508-647-7000
Fax: 508-647-7101
E-mail: info@mathworks.com
Web: www.mathworks.com

It is important for the reader to understand what versions of the software or what toolboxes are used with this text. The book was written using MATLAB Versions 6.5 and 7. We made some use of the MATLAB Statistics Toolbox, Versions 4 and 5. We will refer to the Curve Fitting Toolbox in [Chapter 7](#), where we discuss smoothing. However, this particular toolbox is not needed to use the examples in the book.

To get the most out of this book, readers should have a basic understanding of matrix algebra. For example, one should be familiar with determinants, a matrix transpose, the trace of a matrix, etc. We recommend Strang [1988,

1993] for those who need to refresh their memories on the topic. We do not use any calculus in this book, but a solid understanding of algebra is always useful in any situation. We expect readers to have knowledge of the basic concepts in probability and statistics, such as random samples, probability distributions, hypothesis testing, and regression.

1.3 A Few Words About Notation

In this section, we explain our notation and font conventions. MATLAB code will be in Courier New bold font such as this: **function**. To make the book more readable, we will indent MATLAB code when we have several lines of code, and this can always be typed in as you see it in the book.

For the most part, we follow the convention that a vector is arranged as a column, so it has dimensions $p \times 1$.¹ Our data sets will always be arranged in a matrix of dimension $n \times p$, which is denoted as **X**. Here n represents the number of observations we have in our sample, and p is the number of variables or dimensions. Thus, each row corresponds to a p -dimensional observation or data point. The ij -th element of **X** will be represented by x_{ij} . For the most part, the subscript i refers to a row in a matrix or an observation, and a subscript j references a column in a matrix or a variable. What is meant by this will be clear from the text.

In many cases, we might need to center our observations before we analyze them. To make the notation somewhat simpler later on, we will use the matrix **X_c** to represent our centered data matrix, where each row is now centered at the origin. We calculate this matrix by first finding the mean of each column of **X** and then subtracting it from each row. The following code will calculate this in MATLAB:

```
% Find the mean of each column.
[n,p] = size(X);
xbar = mean(X);
% Create a matrix where each row is the mean
% and subtract from X to center at origin.
Xc = X - repmat(xbar,n,1);
```

¹The notation $m \times n$ is read “ m by n ,” and it means that we have m rows and n columns in an array. It will be clear from the context whether this indicates matrix dimensions or multiplication.

1.4 Data Sets Used in the Book

In this section, we describe the main data sets that will be used throughout the text. Other data sets will be used in the exercises and in some of the examples. This section can be set aside and read as needed without any loss of continuity. Please see [Appendix C](#) for detailed information on all data sets included with the text.

1.4.1 Unstructured Text Documents

The ability to analyze free-form text documents (e.g., Internet documents, intelligence reports, news stories, etc.) is an important application in computational statistics. We must first encode the documents in some numeric form in order to apply computational methods. The usual way this is accomplished is via a term-document matrix, where each row of the matrix corresponds to a word in the lexicon, and each column represents a document. The elements of the term-document matrix contain the number of times the i -th word appears in j -th document [Manning and Schütze, 2000; Charniak, 1996]. One of the drawbacks to this type of encoding is that the order of the words is lost, resulting in a loss of information [Hand, Mannila and Smyth, 2001].

We now present a new method for encoding unstructured text documents where the order of the words is accounted for. The resulting structure is called the bigram proximity matrix (BPM).

Bigram Proximity Matrices

The *bigram proximity matrix* (BPM) is a nonsymmetric matrix that captures the number of times word pairs occur in a section of text [Martinez and Wegman, 2002a; 2002b]. The BPM is a square matrix whose column and row headings are the alphabetically ordered entries of the lexicon. Each element of the BPM is the number of times word i appears immediately before word j in the unit of text. The size of the BPM is determined by the size of the lexicon created by alphabetically listing the unique occurrences of the words in the corpus. In order to assess the usefulness of the BPM encoding we had to determine whether or not the representation preserves enough of the semantic content to make them separable from BPMs of other thematically unrelated collections of documents.

We must make some comments about the lexicon and the pre-processing of the documents before proceeding with more information on the BPM and the data provided with this book. All punctuation *within* a sentence, such as commas, semi-colons, colons, etc., were removed. All end-of-sentence punctuation, other than a period, such as question marks and exclamation

points were converted to a period. The period is used in the lexicon as a word, and it is placed at the beginning of the alphabetized lexicon.

Other pre-processing issues involve the removal of noise words and stemming. Many natural language processing applications use a shorter version of the lexicon by excluding words often used in the language [Kimbrell, 1988; Salton, Buckley and Smith, 1990; Frakes and Baeza-Yates, 1992; Berry and Browne, 1999]. These words, usually called *stop words*, are said to have low informational content and thus, in the name of computational efficiency, are deleted. Not all agree with this approach [Witten, Moffat and Bell, 1994].

Taking the denoising idea one step further, one could also stem the words in the denoised text. The idea is to reduce words to their stem or root to increase the frequency of key words and thus enhance the discriminatory capability of the features. Stemming is routinely applied in the area of information retrieval (IR). In this application of text processing, stemming is used to enhance the performance of the IR system, as well as to reduce the total number of unique words and save on computational resources. The stemmer we used to pre-process the text documents is the Porter stemmer [Baeza-Yates and Ribero-Neto, 1999; Porter, 1980]. The Porter stemmer is simple; however, its performance is comparable with older established stemmers.

We are now ready to give an example of the BPM. The BPM for the sentence or text stream,

“The wise young man sought his father in the crowd.”

is shown in Table 1.1. We see that the matrix element located in the third row (*his*) and the fifth column (*father*) has a value of one. This means that the pair of words *his father* occurs once in this unit of text. It should be noted that in most cases, depending on the size of the lexicon and the size of the text stream, the BPM will be very sparse.

TABLE 1.1

Example of a BPM

	.	crowd	his	in	father	man	sought	the	wise	young
.										
crowd	1									
his					1					
in								1		
father				1						
man							1			
sought			1							
the		1							1	
wise										1
young						1				

Note that the zeros are left out for ease of reading.

By preserving the word ordering of the discourse stream, the BPM captures a substantial amount of information about meaning. Also, by obtaining the individual counts of word co-occurrences, the BPM captures the ‘intensity’ of the discourse’s theme. Both features make the BPM a suitable tool for capturing meaning and performing computations to identify semantic similarities among units of discourse (e.g., paragraphs, documents). Note that a BPM is created for each text unit.

One of the data sets included in this book, which was obtained from text documents, came from the Topic Detection and Tracking (TDT) Pilot Corpus (Linguistic Data Consortium, Philadelphia, PA):

```
comp.soft-sys.matlab/Projects/TDT-Pilot/.
```

The TDT corpus is comprised of close to 16,000 stories collected from July 1, 1994 to June 30, 1995 from the Reuters newswire service and CNN broadcast news transcripts. A set of 25 events are discussed in the complete TDT Pilot Corpus. These 25 topics were determined first, and then the stories were classified as either belonging to the topic, not belonging, or somewhat belonging (*Yes*, *No*, or *Brief*, respectively).

In order to meet the computational requirements of available computing resources, a subset of the TDT corpus was used. A total of 503 stories were chosen that includes 16 of the 25 events. See Table 1.2 for a list of topics. The 503 stories chosen contain only the *Yes* or *No* classifications. This choice stems from the need to demonstrate that the BPM captures enough meaning to make a correct or incorrect topic classification choice.

TABLE 1.2
List of 16 Topics

Topic Number	Topic Description	Number of Documents Used
4	Cessna on the White House	14
5	Clinic Murders (Salvi)	41
6	Comet into Jupiter	44
8	Death of N. Korean Leader	35
9	DNA in OJ Trial	29
11	Hall’s Copter in N. Korea	74
12	Humble, TX Flooding	16
13	Justice-to-be Breyer	8
15	Kobe, Japan Quake	49
16	Lost in Iraq	30
17	NYC Subway Bombing	24
18	Oklahoma City Bombing	76
21	Serbians Down F-16	16
22	Serbs Violate Bihac	19
24	US Air 427 Crash	16
25	WTC Bombing Trial	12

There were 7,146 words in the lexicon after denoising and stemming, so each BPM has $7,146^2$ elements. This is very high dimensional data ($7,146^2$ dimensions). We can apply several EDA methods that require the interpoint distance matrix only and not the original data (i.e., BPMs). Thus, we only include the interpoint distance matrices for different measures of semantic distance: IRad, Ochiai, simple matching, and L_1 . It should be noted that the match and Ochiai measures started out as similarities (large values mean the observations are similar), and were converted to distances for use in the text. See [Appendix A](#) for more information on these distances and Martinez [2002] for other choices, not included here. Table 1.3 gives a summary of the BPM data we will be using in subsequent chapters.

TABLE 1.3
Summary of the BPM Data

Distance	Name of File
IRad	iradbpm
Ochiai	ochiaibpm
Match	matchbpm
L_1 Norm	L1bpm

One of the issues we might want to explore with these data is dimensionality reduction so further processing can be accomplished, such as clustering or supervised learning. We would also be interested in visualizing the data in some manner to determine whether or not the observations exhibit some interesting structure. Finally, we might use these data with a clustering algorithm to see how many groups are found in the data, to find latent topics or sub-groups or to see if documents are clustered such that those in one group have the same meaning.

1.4.2 Gene Expression Data

The Human Genome Project completed a map (in draft form) of the human genetic blueprint in 2001 (<http://www.nature.com/genomics/human>), but much work remains to be done in understanding the functions of the genes and the role of proteins in a living system. The area of study called *functional genomics* addresses this problem, and one of its main tools is DNA *microarray* technology [Sebastiani, et al., 2003]. This technology allows data to be collected on multiple experiments and provides a view of the genetic activity (for thousands of genes) for an organism.

We now provide a brief introduction to the terminology used in this area. The reader is referred to Sebastiani, et al. [2003] or Griffiths, et al. [2000] for more detail on the unique statistical challenges and the underlying biological and technical foundation of genetic analysis. As most of us are aware from

introductory biology, organisms are made up of cells, and the nucleus of each cell contains DNA (deoxyribonucleic acid). DNA instructs the cells to produce proteins and how much protein to produce. Proteins participate in most of the functions living things perform. Segments of DNA are called *genes*. The *genome* is the complete DNA for an organism, and it contains the genetic code needed to create a unique life. The process of gene activation is called *gene expression*, and the expression level provides a value indicating the number of intermediary molecules (messenger ribonucleic acid and transfer ribonucleic acid) created in this process.

Microarray technology can simultaneously measure the relative gene expression level of thousands of genes in tissue or cell samples. There are two main types of microarray technology: cDNA microarrays and synthetic oligonucleotide microarrays. In both of these methods, a target (extracted from tissue or cell) is hybridized to a probe (genes of known identity or small sequences of DNA). The target is tagged with fluorescent dye before being hybridized to the probe, and a digital image is formed of the chemical reaction. The intensity of the signal then has to be converted to a quantitative value from the image. As one might expect, this involves various image processing techniques, and it could be a major source of error.

A data set containing gene expression levels has information on genes (rows of the matrix) from several experiments (columns of the matrix). Typically, the columns correspond to patients, tumors, time steps, etc. We note that with the analysis of gene expression data, either the rows (genes) or columns (experiments/samples) could correspond to the dimensionality (or sample size), depending on the goal of the analysis. Some of the questions that might be addressed through this technology include:

- What genes are expressed (or not expressed) in a tumor cell versus a normal cell?
- Can we predict the best treatment for a cancer?
- Are there genes that characterize a specific tumor?
- Are we able to cluster cells based on their gene expression level?
- Can we discover sub-classes of cancer or tumors?

For more background information on gene expression data, we refer the reader to Schena, et al. [1995], Chee, et al. [1996], and Lander [1999]. Many gene expression data sets are freely available on the internet, and there are also many articles on the statistical analysis of this type of data. We refer the interested reader to a recent issue of *Statistical Science* (Volume 18, Number 1, February 2003) for a special section on microarray analysis. One can also go to the *Proceedings of the National Academy of Science* website (<http://www.pnas.org>) for articles, many of which have the data available for download. We include three gene expression data sets with this book, and we describe them below.

Yeast Data Set

This data set was originally described in Cho, et al. [1998], and it showed the gene expression levels of around 6000 genes over two cell cycles and five phases. The two cell cycles provide 17 time points (columns of the matrix). The subset of the data we provide was obtained by Yeung and Ruzzo [2001] and is available at

<http://www.cs.washington.edu/homes/kayee/model>.

A full description of the process they used to get the subset can also be found there. First, they extracted all genes that were found to peak in only one of the five phases; those that peaked in multiple phases were not used. Then they removed any rows with negative entries, yielding a total of 384 genes.

The data set is called **yeast.mat**, and it contains two variables: **data** and **classlabs**. The **data** matrix has 384 rows and 17 columns. The variable **classlabs** is a vector containing 384 class labels for the genes indicating whether the gene peaks in phase 1 through phase 5.

Leukemia Data Set

The **leukemia** data set was first discussed in Golub, et al., [1999], where the authors measured the gene expressions of human acute leukemia. Their study included prediction of the type of leukemia using supervised learning and the discovery of new classes of leukemia via unsupervised learning. The motivation for this work was to improve cancer treatment by distinguishing between sub-classes of cancer or tumors. The author's website

<http://www.genome.wi.mit.edu/MPR>

has more information about their methods and procedure, and the full data set is available there.

They first classified the leukemias into two groups: (1) those that arise from lymphoid precursors or (2) from myeloid precursors. The first one is called acute lymphoblastic leukemia (ALL), and the second is called acute myeloid leukemia (AML). The distinction between these two classes of leukemia is well known, but a single test to sufficiently establish a diagnosis does not exist [Golub, et al., 1999]. As one might imagine, a proper diagnosis is critical to successful treatment and to avoid unnecessary toxicities. The authors turned to microarray technology and statistical pattern recognition to address this problem.

Their initial data set had 38 bone marrow samples taken at the time of diagnosis; 27 came from patients with ALL, and 11 patients had AML. They used oligonucleotide microarrays containing probes for 6,817 human genes to obtain the gene expression information. Their first goal was to construct a classifier using the gene expression values that would predict the type of leukemia. So, one could consider this as building a classifier where the observations have 6,817 dimensions, and the sample size is 38. They had to reduce the dimensionality, so they chose the 50 genes that have the highest

correlation with the class of leukemia. They used an independent test set of leukemia samples to evaluate the classifier. This set of data consists of 34 samples, where 24 of them came from bone marrow and 10 came from peripheral blood samples. It also included samples from children and from different laboratories using different protocols.

They also looked at class discovery or unsupervised learning, where they wanted to see if the patients could be clustered into two groups corresponding to the types of leukemia. They used the method called self-organizing maps ([Chapter 3](#)), employing the full set of 6,817 genes. Another aspect of class discovery is to look for subgroups within known classes. For example, the patients with ALL can be further subdivided into patients with B-cell or T-cell lineage.

We decided to include only the 50 genes, rather than the full set. The **leukemia.mat** file has four variables. The variable **leukemia** has 50 genes (rows) and 72 patients (columns). The first 38 columns correspond to the initial training set of patients, and the rest of the columns contain data for the independent testing set. The variables **btcell** and **cancertype** are cell arrays of strings containing the label for B-cell, T-cell, or NA and ALL or AML, respectively. Finally, the variable **geneinfo** is a cell array where the first column provides the gene description, and the second column contains the gene number.

Example 1.1

We show a plot of the 50 genes in [Figure 1.1](#), but only the first 38 samples (i.e., columns) are shown. This is similar to Figure 3B in Golub, et al., [1999]. We standardized each gene, so the mean across each row is 0 and the standard deviation is 1. The first 27 columns of the picture correspond to ALL leukemia, and the last 11 columns pertain to the AML leukemia. We can see by the color that the first 25 genes tend to be more highly expressed in ALL, while the last 25 genes are highly expressed in AML. The MATLAB code to construct this plot is given below.

```
% First standardize the data such that each row
% has mean 0 and standard deviation 1.
load leukemia
x = leukemia(:,1:38);
[n,p] = size(x);
y = zeros(n,p);
for i = 1:n
    sig = std(x(i,:));
    mu = mean(x(i,:));
    y(i,:) = (x(i,:) - mu) / sig;
end
% Now do the image of the data.
pcolor(y)
colormap(gray(256))
```

```

colorbar
title('Gene Expression for Leukemia')
xlabel('ALL (1-27) or AML (28-38)')
ylabel('Gene')

```

The results shown in Figure 1.1 indicate that we might be able to distinguish between AML and ALL leukemia using these data.

□

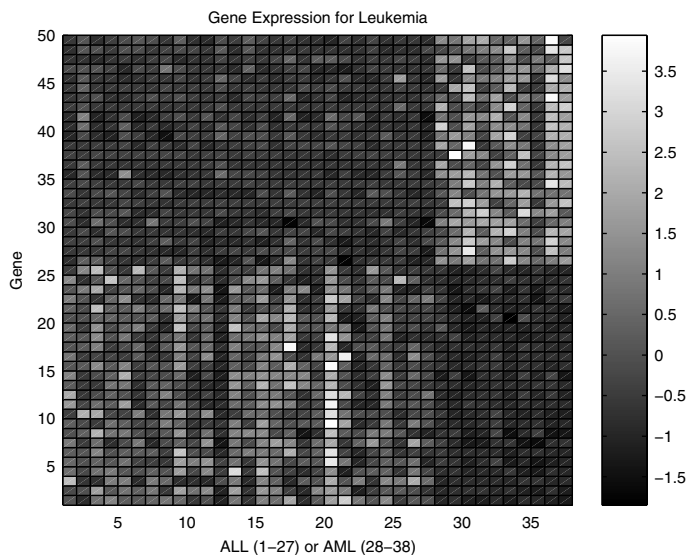


FIGURE 1.1

This shows the gene expression for the **leukemia** data set. Each row corresponds to a gene, and each column corresponds to a cancer sample. The rows have been standardized such that the mean is 0 and the standard deviation is 1. We can see that the ALL leukemia is highly expressed in the first set of 25 genes, and the AML leukemia is highly expressed in the second set of 25 genes.

Lung Data Set

Traditionally, the classification of lung cancer is based on clinicopathological features. An understanding of the molecular basis and a possible molecular classification of lung carcinomas could yield better therapies targeted to the type of cancer, superior prediction of patient treatment, and the identification of new targets for chemotherapy. We provide two data sets that were originally downloaded from <http://www.genome.mit.edu/MPR/lung> and described in Bhattacharjee, et al. [2001]. The authors applied hierarchical and probabilistic clustering to find subclasses of lung adenocarcinoma, and they showed the diagnostic potential of analyzing gene expression data by

demonstrating the ability to separate primary lung adenocarcinomas from metastases of extra-pulmonary origin.

A preliminary classification of lung carcinomas comprises two groups: small-cell lung carcinomas (SCLC) or nonsmall-cell lung carcinomas (NSCLC). The NSCLC category can be further subdivided into 3 groups: adenocarcinomas (AD), squamous cell carcinomas (SQ), and large-cell carcinomas (COID). The most common type is adenocarcinomas. The data were obtained from 203 specimens, where 186 were cancerous and 17 were normal lung. The cancer samples contained 139 lung adenocarcinomas, 21 squamous cell lung carcinomas, 20 pulmonary carcinoids, and 6 small-cell lung carcinomas. This is called Dataset A in Bhattacharjee, et al. [2001]; the full data set included 12,600 genes. The authors reduced this to 3,312 by selecting the most variable genes, using a standard deviation threshold of 50 expression units. We provide these data in **lungA.mat**. This file includes two variables: **lungA** and **labA**. The variable **lungA** is a 3312×203 matrix, and **labA** is a vector containing the 203 class labels.

The authors also looked at adenocarcinomas separately trying to discover subclasses. To this end, they separated the 139 adenocarcinomas and the 17 normal samples and called it Dataset B. They also took fewer gene transcript sequences for this data set by selecting only 675 genes according to other statistical pre-processing steps. These data are provided in **lungB.mat**, which contains two variables: **lungB** (675×156) and **labB** (156 class labels). We summarize these data sets in Table 1.4.

TABLE 1.4
Description of Lung Cancer Data Set

Cancer Type	Label	Number of Data Points
<i>Dataset A (lungA.mat): 3,312 rows, 203 columns</i>		
Nonsmall cell lung carcinomas		
Adenocarcinomas	AD	139
Pulmonary carcinoids	COID	20
Squamous cell	SQ	21
Normal	NL	17
Small-cell lung carcinomas	SCLC	6
<i>Dataset B (lungB.mat): 675 rows, 156 columns</i>		
Adenocarcinomas	AD	139
Normal	NL	17

For those who need to analyze gene expression data, we recommend the Bioinformatics Toolbox from The MathWorks. The toolbox provides an integrated environment for solving problems in genomics and proteomics, genetic engineering, and biological research. Some capabilities include the ability to calculate the statistical characteristics of the data, to manipulate sequences, to construct models of biological sequences using Hidden Markov Models, and to visualize microarray data.

1.4.3 Oronsay Data Set

This data set consists of particle size measurements originally presented in Timmins [1981] and analyzed by Olbricht [1982], Fieller, Gilbertson & Olbricht [1984], and Fieller, Flenley and Olbricht [1992]. An extensive analysis from a graphical EDA point of view was conducted by Wilhelm, Wegman and Symanzik [1999]. The measurement and analysis of particle sizes is often used in archaeology, fuel technology (droplets of propellant), medicine (blood cells), and geology (grains of sand). The usual objective is to determine the distribution of particle sizes because this characterizes the environment where the measurements were taken or the process of interest.

The Oronsay particle size data were gathered for a geological application, where the goal was to discover different characteristics between dune sands and beach sands. This characterization would be used to determine whether or not midden sands were dune or beach. The middens were near places where prehistoric man lived, and geologists are interested in whether these middens were beach or dune because that would be an indication of how the coastline has shifted.

There are 226 samples of sand, with 77 belonging to an unknown type of sand (from the middens) and 149 samples of known type (beach or dune). The known samples were taken from *Cnoc Coig* (CC - 119 observations, 90 beach and 29 dune) and *Caisteal nan Gilleann* (CG - 30 observations, 20 beach and 10 dune). See Wilhelm, Wegman and Symanzik [1999] for a map showing these sites on Oronsay island. This reference also shows a more detailed classification of the sands based on transects and levels of sand.

Each observation is obtained in the following manner. Approximately 60g or 70g of sand is put through a stack of 11 sieves of sizes 0.063mm, 0.09mm, 0.125mm, 0.18mm, 0.25mm, 0.355mm, 0.5mm, 0.71mm, 1.0mm, 1.4mm, and 2.0mm. The sand that remains on each of the sieves is weighed, along with the sand that went through completely. This yields 12 weight measurements, and each corresponds to a class of particle size. Note that there are two extreme classes: particle sizes less than 0.063mm (what went through the smallest sieve) and particle sizes larger than 2.0mm (what is in the largest sieve).

Flenley and Olbricht [1993] consider the classes as outlined above, and they apply various multivariate and exploratory data analysis techniques such as principal component analysis and projection pursuit. The **oronsay** data set was downloaded from:

<http://www.galaxy.gmu.edu/papers/oronsay.html>

More information on the original data set can be found at this website. We chose to label observations first with respect to midden, beach or dune (in variable **beachdune**):

- Class 0: midden (77 observations)
- Class 1: beach (110 observations)

- Class 2: dune (39 observations)

We then classify observations according to the sampling site (in variable **midden**), as follows

- Class 0: midden (77 observations)
- Class 1: *Cnoc Coig* - CC (119 observations)
- Class 2: *Caisteal nan Gilleann* - CG (30 observations)

The data set is in the **oronsay.mat** file. The data are in a 226×12 matrix called **oronsay**, and the data are in raw format; i.e., untransformed and unstandardized. Also included is a cell array of strings called **labcol** that contains the names (i.e., sieve sizes) of the columns.

1.4.4 Software Inspection

The data described in this section were collected in response to efforts for process improvement in software testing. Many systems today rely on complex software that might consist of several modules programmed by different programmers, so ensuring that the software works correctly and as expected is important.

One way to test the software is by inspection, where software engineers inspect the code in a formal way. First they look for inconsistencies, logical errors, etc., and then they all meet with the programmer to discuss what they perceive as defects. The programmer is familiar with the code and can help determine whether or not it is a defect in the software.

The data are saved in a file called **software**. The variables are normalized by the size of the inspection (the number of pages or SLOC – single lines of code). The file **software.mat** contains the preparation time in minutes (**prepage**, **prepsloc**), the total work hours in minutes for the meeting (**mtgsloc**), and the number of defects found (**defpage**, **defsloc**). Software engineers and managers would be interested in understanding the relationship between the inspection time and the number of defects found. One of the goals might be to find an optimal time for inspection, where one gets the most payoff (number of defects found) for the amount of time spent reviewing the code. We show an example of these data in [Figure 1.2](#). The defect types include compatibility, design, human-factors, standards, and others.

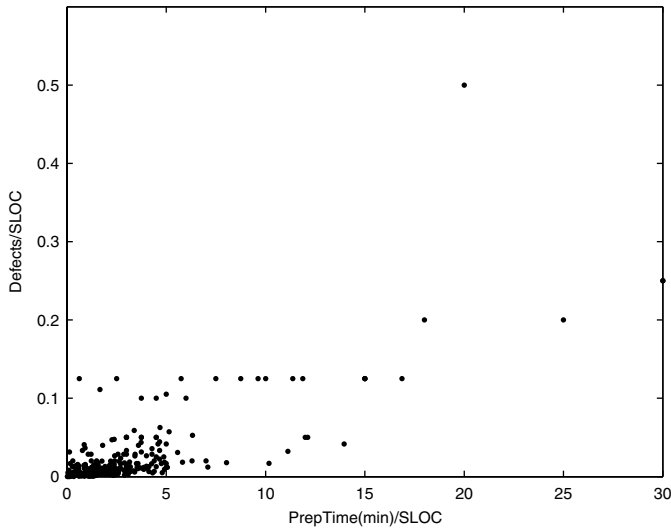


FIGURE 1.2
This is a scatterplot of the software inspection data. The relationship between the variables is difficult to see.

1.5 Transforming Data

In many real-world applications, the data analyst will have to deal with raw data that are not in the most convenient form. The data might need to be re-expressed to produce effective visualization or an easier, more informative analysis. Some of the types of problems that can arise include data that exhibit nonlinearity or asymmetry, contain outliers, change spread with different levels, etc. We can transform the data by applying a single mathematical function to all of the observations.

In the first sub-section below, we discuss the general power transformations that can be used to change the shape of the data distribution. This arises in situations when we are concerned with formal inference methods where the shape of the distribution is important (e.g., statistical hypothesis testing or confidence intervals). In EDA, we might want to change the shape to facilitate visualization, smoothing, and other analyses. Next we cover linear transformations of the data that leave the shape alone. These are typically changes in scale and origin and can be important in dimensionality reduction, clustering, and visualization.

1.5.1 Power Transformations

A **transformation** of a set of data points $x_1, x_2, \dots, x_n$ is a function T that substitutes each observation x_i with a new value $T(x_i)$ [Emerson and Stoto, 1983]. Transformations should have the following desirable properties:

1. The order of the data is preserved by the transformation. Because of this, statistics based on order, such as medians are preserved; i.e., medians are transformed to medians.
2. They are continuous functions guaranteeing that points that are close together in raw form are also close together using their transformed values, relative to the scale used.
3. They are smooth functions that have derivatives of all orders, and they are specified by elementary functions.

Some common transformations include taking roots (square root, cube root, etc.), finding reciprocals, calculating logarithms, and raising variables to positive integral powers. These transformations provide adequate flexibility for most situations in data analysis.

Example 1.2

This example uses the software inspection data shown in [Figure 1.2](#). We see that the data are skewed, and the relationship between the variables is difficult to understand. We apply a log transform to both variables using the following MATLAB code, and show the results in [Figure 1.3](#).

```
load software
% First transform the data.
X = log(prepsloc);
Y = log(defsloc);
% Plot the transformed data.
plot(X,Y, '.')
xlabel('Log PrepTime/SLOC')
ylabel('Log Defects/SLOC')
```

We now have a better idea of the relationship between these two variables, which will be examined further in [Chapter 7](#).

□

Some transformations of the data may lead to insights or discovery of structures that we might not otherwise see. However, as with any analysis, we should be careful about creating something that is not really there, but is just an artifact of the processing. Thus, in any application of EDA, the analyst should go back to the subject area and consult domain experts to verify and help interpret the results.

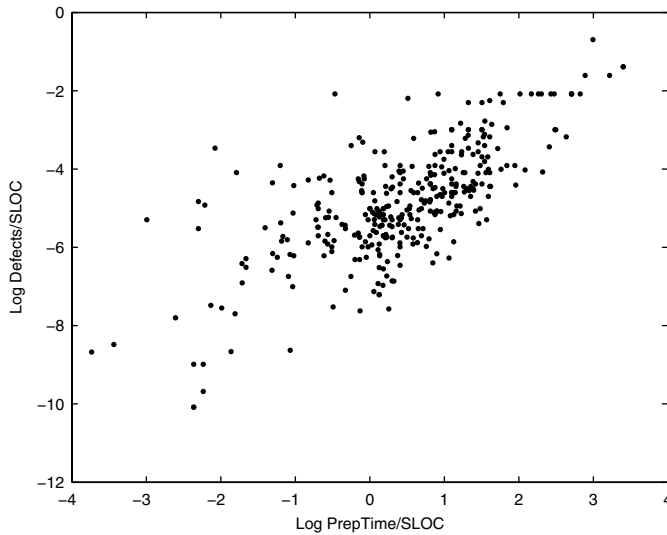


FIGURE 1.3

Each variate was transformed using the logarithm. The relationship between preparation time per SLOC and number of defects found per SLOC is now easier to see.

1.5.2 Standardization

If the variables are measurements along a different scale or if the standard deviations for the variables are different from one another, then one variable might dominate the distance (or some other similar calculation) used in the analysis. We will make extensive use of interpoint distances throughout the text in applications such as clustering, multidimensional scaling, and nonlinear dimensionality reduction. We discuss several 1-D standardization methods below. However, we note that in some multivariate contexts, the 1-D transformations may be applied to each variable (i.e., on the column of $\mathbf{X}$) separately.

Transformation Using the Standard Deviation

The first standardization we discuss is called the sample *z-score*, and it should be familiar to most readers who have taken an introductory statistics class. The transformed variates are found using

$$z = \frac{(x - \bar{x})}{s}, \quad (1.1)$$

where x is the original observed data value, $\bar{x}$ is the sample mean, and s is the sample standard deviation. In this standardization, the new variate z will have a mean of zero and a variance of one.

When the z -score transformation is used in a clustering context, it is important that it be applied in a global manner across all observations. If standardization is done within clusters, then false and misleading clustering solutions can result [Milligan and Cooper, 1988].

If we do not center the data at zero by removing the sample mean, then we have the following

$$z = \frac{x}{s}. \quad (1.2)$$

This transformed variable will have a variance of one and a transformed mean equal to $\bar{x}/s$. The standardizations in Equations 1.1 and 1.2 are linear functions of each other, so Euclidean distances (see [Appendix A](#)) calculated on data that have been transformed using the two formulas result in identical dissimilarity values.

For robust versions of Equations 1.1 and 1.2, we can substitute the median and the interquartile range for the sample mean and sample standard deviation respectively. This will be explored in the exercises.

Transformation Using the Range

Instead of dividing by the standard deviation, as above, we can use the range of the variable as the divisor. This yields the following two forms of standardization

$$z = \frac{x}{\max(x) - \min(x)}, \quad (1.3)$$

and

$$z = \frac{x - \min(x)}{\max(x) - \min(x)}. \quad (1.4)$$

The standardization in Equation 1.4 is bounded by zero and one, with at least one observed value at each of the end points. The transformed variate given by Equation 1.3 is a linear function of the one determined by Equation 1.4, so data standardized using these transformations will result in identical Euclidean distances.

1.5.3 Sphering the Data

This type of standardization called *sphering* pertains to multivariate data, and it serves a similar purpose as the 1-D standardization methods given above. The transformed variables will have a p -dimensional mean of $\mathbf{0}$ and a covariance matrix given by the identity matrix.

We start off with the p -dimensional sample mean given by

$$\bar{\mathbf{x}} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i.$$

We then find the sample covariance matrix given by the following

$$\mathbf{S} = \frac{1}{n-1} \sum_{i=1}^n (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T,$$

where we see that the covariance matrix can be written as the sum of n matrices. Each of these rank one matrices is the outer product of the centered observations [Duda and Hart, 1973].

We sphere the data using the following transformation

$$\mathbf{Z}_i = \Lambda^{-1/2} \mathbf{Q}^T (\mathbf{x}_i - \bar{\mathbf{x}}) \quad i = 1, \dots, n,$$

where the columns of $\mathbf{Q}$ are the eigenvectors obtained from $\mathbf{S}$, Λ is a diagonal matrix of corresponding eigenvalues, and $\mathbf{x}_i$ is the i -th observation.

Example 1.3

We now provide the MATLAB code to obtain sphered data. First, we generate 2-D multivariate normal random variables that have the following parameters:

$$\mu = \begin{bmatrix} -2 \\ 2 \end{bmatrix},$$

and

$$\Sigma = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix},$$

where Σ is the covariance matrix. A scatterplot of these data is shown in Figure 1.4 (top).

```
% First generate some 2-D multivariate normal
% random variables, with mean MU and
% covariance SIGMA. This uses a Statistics
% Toolbox function.
n = 100;
mu = [-2, 2];
sigma = [1, .5; .5, 1];
X = mvnrnd(mu, sigma, n);
plot(X(:,1), X(:,2), '.')
```

We now apply the steps to sphere the data, and show the transformed data in Figure 1.4 (bottom).

```
% Now sphere the data.
xbar = mean(X);
% Get the eigenvectors and eigenvalues of the
% covariance matrix.
[V,D] = eig(cov(X));
% Center the data.
Xc = X - ones(n,1)*xbar;
% Sphere the data.
Z = ((D)^(-1/2)*V'*Xc)';
plot(Z(:,1), Z(:,2), '.')
```

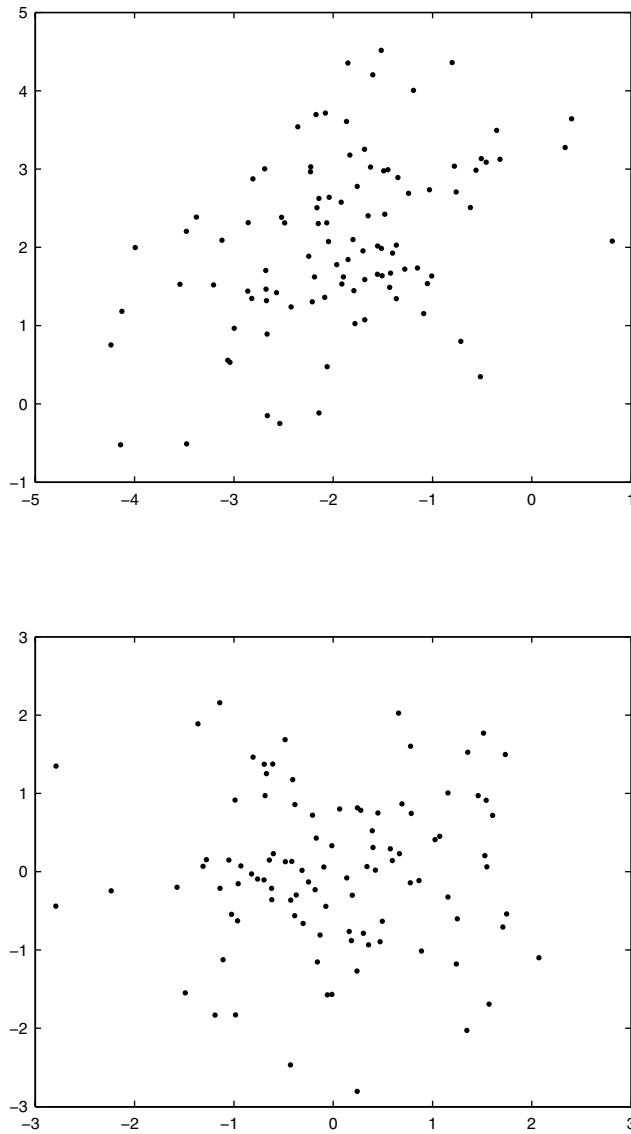
By comparing these two plots, we see that the transformed data are sphered and are now centered at the origin.

□

1.6 Further Reading

There are several books that will provide the reader with more information and other perspectives on EDA. Most of these books do not offer software and algorithms, but they are excellent resources for the student or practitioner of exploratory data analysis.

As we stated in the beginning of this chapter, the seminal book on EDA is Tukey [1977], but the text does not include the more up-to-date view based on current computational resources and methodology. Similarly, the short book on EDA by Hartwig and Dearing [1979] is an excellent introduction to the topic and a quick read, but it is somewhat dated. For the graphical approach, the reader is referred to du Toit, Steyn and Stumpf [1986], where the authors use SAS to illustrate the ideas. They include other EDA methods such as multidimensional scaling and cluster analysis. Hoaglin, Mosteller

**FIGURE 1.4**

The top figure shows a scatterplot of the 2-D multivariate normal random variables. Note that these are not centered at the origin, and the cloud is not spherical. The sphered data are shown in the bottom panel. We see that they are now centered at the origin with a spherical spread. This is similar to the z-score standardization in 1-D.

and Tukey [1983] edited an excellent book on robust and exploratory data analysis. It includes several chapters on transforming data, and we recommend the one by Emerson and Stoto [1983]. The chapter includes a discussion of power transformations, as well as plots to assist the data analyst in choosing an appropriate one.

For a more contemporary resource that explains data mining approaches, of which EDA is a part, Hand, Mannila and Smyth [2001] is highly recommended. It does not include computer code, but it is very readable. The authors cover the major parts of data mining: EDA, descriptive modeling, classification and regression, discovering patterns and rules, and retrieval by content. Finally, the reader could also investigate the book by Hastie, Tibshirani and Friedman [2001]. These authors cover a wide variety of topics of interest to exploratory data analysts, such as clustering, nonparametric probability density estimation, multidimensional scaling, and projection pursuit.

As was stated previously, EDA is sometimes defined as an attitude of flexibility and discovery in the data analysis process. There is an excellent article by Good [1982] outlining the philosophy of EDA, where he states that “EDA is more an art, or even a bag of tricks, than a science.” While we do not think there is anything “tricky” about the EDA techniques, it is somewhat of an art in that the analyst must try various methods in the discovery process, keeping an open mind and being prepared for surprises! Finally, other summaries of EDA were written by Diaconis [1985] and Weihs [1993]. Weihs describes EDA mostly from a graphical viewpoint and includes descriptions of dimensionality reduction, grand tours, prediction models, and variable selection. Diaconis discusses the difference between exploratory methods and the techniques of classical mathematical statistics. In his discussion of EDA, he considers Monte Carlo techniques such as the bootstrap [Efron and Tibshirani, 1993].

Exercises

- 1.1 What is exploratory data analysis? What is confirmatory data analysis? How do these analyses fit together?
- 1.2 Repeat Example 1.1 using the remaining columns (39 – 72) of the **leukemia** data set. Does this follow the same pattern as the others?
- 1.3 Repeat Example 1.1 using the **lungB** gene expression data set. Is there a pattern?
- 1.4 Generate some 1-D normally distributed random variables with $\mu = 5$ and $\sigma = 2$ using **normrnd** or **randn** (must transform the results to have the required mean and standard deviation if you use this function). Apply the various standardization procedures described in

- this chapter and verify the comments regarding the location and spread of the transformed variables.
- 1.5 Write MATLAB functions that implement the standardizations mentioned in this chapter.
 - 1.6 Using the **mvnrnd** function (see Example 1.3), generate some nonstandard bivariate normal random variables. Sphere the data and verify that the resulting sphered data have mean 0 and identity covariance matrix using the MATLAB functions **mean** and **cov**.
 - 1.7 We will discuss the quartiles and the interquartile range in [Chapter 9](#), but for now look at the MATLAB **help** files on the **iqr** and **median** functions. We can use these robust measures of location and spread to transform our variables. Using Equations 1.1 and 1.2, substitute the median for the sample mean $\bar{x}$ and the interquartile range for the sample standard deviation s . Write a MATLAB function that does this and try it out on the same data you generated in problem 1.4.
 - 1.8 Generate $n = 2$ normally distributed random variables. Find the Euclidean distance between the points after they have been transformed first using Equation 1.1 and then Equation 1.2. Are the distances the same? Hint: Use the **pdist** function from the Statistics Toolbox.
 - 1.9 Repeat problem 1.8 using the standardizations given by Equations 1.3 and 1.4.
 - 1.10 Generate $n = 100$ uniform 1-D random variables using the **rand** function. Construct a histogram using the **hist** function. Now transform the data by taking the logarithm (use the **log** function). Construct a histogram of these transformed values. Did the shape of the distribution change? Comment on the results.
 - 1.11 Try the following transformations on the software data:

$$T(x) = \log(1 + \sqrt{x})$$

$$T(x) = \log(\sqrt{x})$$

Construct a scatterplot and compare with the results in Example 1.2.