1 Data Analysis Methodology

Suppose you inherited the database in Table 1.1 and needed to find out what could be learned from it—fast. Say your boss entered your office and said, "Here's some software project data your predecessor collected. Is there anything interesting about it? I'd like you to present the results at next week's management meeting." Given that it usually takes a number of years to collect enough software project data to analyze, plus the software industry's high job turnover rate, and more often than not, you probably will be analyzing data that was collected by others.

What is this data? What do the abbreviations mean? What statistical methods should you use? What should you do first? Calm down and read on. After eight years of collecting, validating, analyzing, and benchmarking software projects, I've written the book that I wish had been available the day I was told to "find something interesting" about the European Space Agency software project database.

In this chapter, I will share with you my data analysis methodology. Each step is demonstrated using the software project data in Table 1.1. You do not need to understand statistics to follow the "recipe" in Sidebar 1.1. I simply

TABLE 1.1
Software Project Data

id	*effort*	*size*	*app*	*telonuse*	*t13*	*t14*
2	7871	647	TransPro	No	4	4
3	845	130	TransPro	No	4	4
5	21272	1056	CustServ	No	3	2
6	4224	383	TransPro	No	5	4
8	7320	209	TransPro	No	4	2
9	9125	366	TransPro	No	3	2
15	2565	249	InfServ	No	2	4
16	4047	371	TransPro	No	3	3
17	1520	211	TransPro	No	3	3
18	25910	1849	TransPro	Yes	3	3
19	37286	2482	TransPro	Yes	3	1
21	11039	292	TransPro	No	4	2
25	10447	567	TransPro	Yes	2	2
26	5100	467	TransPro	Yes	2	3
27	63694	3368	TransPro	No	4	2
30	1745	185	InfServ	No	4	5
31	1798	387	CustServ	No	3	3
32	2957	430	MIS	No	3	4
33	963	204	TransPro	No	3	3
34	1233	71	TransPro	No	2	4
38	3850	548	CustServ	No	4	3
40	5787	302	MIS	No	2	4
43	5578	227	TransPro	No	2	3
44	1060	59	TransPro	No	3	3
45	5279	299	InfServ	Yes	3	2
46	8117	422	CustServ	No	3	2
50	1755	193	TransPro	No	2	4
51	5931	1526	InfServ	Yes	4	3
53	3600	509	TransPro	No	4	2
54	4557	583	MIS	No	5	3
55	8752	315	CustServ	No	3	3
56	3440	138	CustServ	No	4	3
58	13700	423	TransPro	No	4	2
61	4620	204	InfServ	Yes	3	2

Sidebar 1.1
Data Analysis Recipe

Ingredients:
As much high-quality data as possible
One package statistical analysis software
A good dose of common sense

Step 1: Validate your data
Step 2: Select the variables and model
Step 3: Perform preliminary analyses (using graphs, tables, correlation and stepwise regression analyses)
Step 4: Build the multi-variable model (using analysis of variance)
Step 5: Check the model
Step 6: Extract the equation

After you fully understand Steps 1 through 6, which are explained in this chapter, read the case studies in Chapters 2 through 5 to gain experience analyzing more complicated databases and to learn how to transform your equations into management implications. See Chapter 5 for an example of how to serve your results to your guests. If you have time, refer to Chapter 6 to learn more about the different statistical methods used in the recipe.

explain what to do, why we do it, how to interpret the statistical output results, and what to watch out for at each stage.

Data Validation

The most important step is data validation. I spend much more time validating data than I do analyzing it. Often, data is not neatly presented to you in one table as it is in this book, but it is in several files that need to be merged and which may include information you do not need or understand. The data may also exist on different pieces of paper.

What do I mean by data validation? In general terms, I mean finding out if you have the right data for your purpose. It is not enough to write a questionnaire and get people to fill it out; you need to have a vision. Like getting the requirement specifications right before starting to develop the software. Specifically, you need to determine if the values for each variable make sense.

Why Do It? You can waste months trying to make sense out of data that was collected without a clear purpose, and without statistical analysis

requirements in mind. It is much better to get a precise idea of exactly what data you have and how much you trust it before you start analyzing. Regardless of whether the data concerns chocolate bar sales, financial indicators, or software projects, the old maxim "garbage in equals garbage out" applies. If you find out that something is wrong with the raw data after you have analyzed it, your conclusions are meaningless. In the best case, you may just have to correct something and analyze it all again. However, if the problem lies with the definition of a variable, it may be impossible to go back and collect the data needed. If you are collecting the data yourself, make sure you ask the right questions the first time. You may not have a second chance.

How to Do It Start off by asking these questions:

- What is this data?
- When was the data collected?
- Why was the data collected?
- Who collected it?
- How did that person ensure that everyone understood the definitions?
- What is the definition of each variable?
- What are the units of measurement of each variable?
- What are the definitions of the values of each variable?

Example The software development project data in Table 1.1 describes 34 COBOL applications running in a mainframe environment at a bank. Most of these applications used other languages in addition to COBOL. The project's manager collected the data upon completion of each project. One person entered all project data into a company database and validated it. The purpose of the data collection was to help manage project portfolios at the bank. Table 1.2 defines the variables. I recommend that you create a table like this for each database you analyze. It is important to be very organized so you can return to your work later and understand it (or leave it for someone else to understand).

Once we understand what the variables are, we need to check that the values make sense. One easy way to do this is to use a data summary function for all variables with numerical values. Example 1.1 shows the number of observations (*Obs*), the average (*Mean*), the standard deviation (*Std*. *Dev*.), the minimum (*Min*), and the maximum (*Max*) value for each variable. For the moment, we are just interested in the number of observations and range of values. If the number of observations is not the same for each variable, this

TABLE 1.2
Variable Definitions

Variable	Full Name	Definition
id	identification number	Each completed project has a unique identification number. (Originally, each project was given a name instead of a number, but I replaced these names for data confidentiality reasons.)
effort	effort	Work carried out by the software supplier from specification until delivery, measured in hours.
size	application size	Function points measured using the Experience method.
app	application type	CustServ = Customer service MIS = Management information system TransPro = Transaction processing InfServ = Information/on-line service
telonuse	Telon use	Telon is a tool that generates code. No = No Telon used Yes = Telon used
t13	staff application knowledge	Knowledge of application domain in project team (supplier and customer): 1 = Very low; team application experience <6 months on average 2 = Low; application experience low; some members have experience; 6-12 months on average 3 = Nominal; application experience good; 1-3 years on average 4 = High; application experience good both at supplier and customer sites; 3-6 years on average; business dynamics known 5 = Very high; both supplier and customer know application area well, including the business; >6 years' average experience
t14	staff tool skills	Experience level of project team (supplier and customer) in regard to development and documentation tools at project kick-off: 1 = Very low; team has no experience in necessary tools; team's average experience <6 months

(*continued*)

TABLE 1.2 (*continued*)

Variable	Full Name	Definition
		2 = Low; tools experience less than average; some members have experience with some tools; 6-12 months on average 3 = Nominal; tools experience good in about half the team; some members know development and documentation tools well; 1-3 years on average 4 = High; most team members know tools well; some members can help others; 3-6 years on average 5 = Very high; team knows all tools well; support available for specific needs of project; >6 years' average experience

means that data is missing. This may be normal as all variables may not have been collected for each project, or it may point to a problem. See if you can find these missing values and add them to the database before you go any further. Also, check to see if the maximum and minimum values make sense. In this case, they do. But if *t13* or *t14* had 7 as a maximum value, we would immediately know there was a problem because by definition, 5 is the highest value possible.

This is also a useful exercise to undertake when someone transfers a very large database to you via the Internet. When it is impossible to check each value individually, check the summary values with the person who sent you the data. I also recommend checking all the variables one-by-one for the first project, the last project, and a few random projects from the middle of the database to make sure nothing got altered during the transfer.

Example 1.1

```
. summarize
```

Variable	Obs	Mean	Std. Dev.	Min	Max
id	34	31.5	17.9059	2	61
effort	34	8734.912	12355.46	845	63694
size	34	578.5882	711.7584	59	3368
t13	34	3.235294	.8548905	2	5
t14	34	2.911765	.9000891	1	5

Next, I tabulate each variable that has words or letters as values. Besides providing valuable information about how many projects are in each category, it is also an easy way to check for spelling mistakes. For example, if there was one observation for *CustSer* and five observations for *CustServ*, you should check if there are really two different categories.

In Examples 1.2 and 1.3, *Freq.* is the number of observations in each category, *Percent* is the percentage of observations in each category, and *Cum.* is the cumulative percentage. We can see that the majority of the applications (about 59%) are transaction processing (*TransPro*) applications. Seven applications used Telon in addition to COBOL. For business presentations, this type of information would look good displayed in a pie chart.

Example 1.2

```
. tabulate app

Application Type    Freq.    Percent      Cum.
CustServ              6       17.65      17.65
MIS                   3        8.82      26.47
TransPro             20       58.82      85.29
InfServ               5       14.71     100.00
Total                34      100.00
```

Example 1.3

```
. tabulate telonuse

Telon Use    Freq.    Percent      Cum.
No             27       79.41      79.41
Yes             7       20.59     100.00
Total          34      100.00
```

What to Watch Out For

- What does a blank mean? (Missing? Don't know? None?)
- What does a 0 mean? (0? Missing? A number close to 0 that has been rounded to 0?)
- If there is an "*Other*" category, what does it include? If the "*Other*" category is extremely diverse, it can't be analyzed as a homogenous category.
- Can you cross-check any variables to see if they make sense? For example, if you have the start date, end date, and duration of a project, you can check if *end date – start date = duration*.

Variable and Model Selection

Once we understand what data we actually have, we need to determine what we can learn from it. What possible relationships could we study? What possible relationships should we study? What relationship will we study first? Your answers to the last two questions will depend on the overall goals of the analysis.

Why Do It? The data may have been collected for a clearly stated purpose. Even so, there might be other interesting relationships to study that occur to you while you are analyzing the data, and which you might be tempted to investigate. However, it is important to decide in advance what you are going to do first and then to complete that task in a meticulous, organized manner. Otherwise, you will find yourself going in lots of different directions, generating lots of computer output, and becoming confused about what you have tried and what you have not tried; in short, you will drown yourself in the data. It is also at this stage that you may decide to create new variables or to reduce the number of variables in your analysis. Variables of questionable validity, variables not meaningfully related to what you want to study, and categorical variable values that do not have a sufficient number of observations should be dropped from the analysis. (See the case studies in Chapters 2 through 5 for examples of variable reduction/modification.) In the following example, we will use all the variables provided in Table 1.1.

Example The smallest number of observations for a categorical variable is 3 for the *MIS* (management information systems) category of the application type (*app*) variable (see Example 1.2). Given that our data set contains 34 observations, I feel comfortable letting *MIS* be represented by three projects. No matter how many observations the database contains, I don't believe it is wise to make a judgment about something represented by less than three projects. This is my personal opinion. Ask yourself this: If the *MIS* category contained only one project and you found in your statistical analysis that the *MIS* category had a significantly higher productivity, would you then conclude that all *MIS* projects in the bank have a high productivity? I would not. If there were two projects, would you believe it? I would not. If there were three projects, would you believe it? Yes, I would in this case. However, if there were 3000 projects in the database, I would prefer for *MIS* to be represented by more than three projects. Feel free to use your own judgment.

Even with this small sample of software project data, we could investigate a number of relationships. We could investigate if any of the factors collected influenced software development effort. Or we could find out which factors influenced software development productivity (i.e., *size/effort*). We could also look at the relationship between application size (*size*) and Telon use (*telonuse*), between size and application type (*app*), or between application type (*app*) and staff application knowledge (*t13*), just to name a few more possibilities. In this example, we will focus on determining which factors affect effort. That is, do size, application type (*app*), Telon use (*telonuse*), staff application knowledge (*t13*), staff tool skills (*t14*), or a combination of these factors have an impact on effort? Is effort a function of these variables? Mathematically speaking, does:

$$effort = f\,(size, app, telonuse, t13, t14)?$$

In this equation, effort is on the left-hand side (LHS) and the other variables are on the right-hand side (RHS). We refer to the LHS variable as the dependent variable and the RHS variables as independent variables.

What to Watch Out For

- To develop a predictive model, make sure that the independent variables are all factors that you know or can predict with reasonable accuracy in advance.
- Category values with less than three observations should be dropped from any multi-variable analysis.

Preliminary Analyses

Before running "blind" statistical tests, I check that the assumptions underlying them are true. In addition, I like to get some first impressions of the data. My objective is not a complete understanding of all possible relationships among all the variables. For example, in Step 2, variable and model selection, I decided that my first goal was to determine which of the variables collected had an influence on effort. To achieve that goal, I follow the steps described in this section before building the multi-variable model (Step 4).

Graphs

Histograms To start, I look at a graph of each numerical variable individually to see how many small values, large values, and medium values

there are, that is, the distribution of each variable. These are also called histograms.

Why Do It? I want to see if the variables are normally distributed. Many statistical techniques assume that the underlying data is normally distributed, so you should check if it is. A normal distribution is also known as a bell-shaped curve. Many of us were graded on such curves at large competitive universities. In a bell-shaped curve, most values fall in the middle, with few very high and very low values. For example, if an exam is graded and the results are fit to a normal distribution (Figure 1.1), most students will get a C. Less students will get a B or a D. And even fewer students will receive an A or an F. The average test score will be the midpoint of the C grade, whether the score is 50, or 90, out of 100. That does not always seem very fair, does it? You can learn more about normal distributions and why they are important in Chapter 6.

How to Do It To create a histogram for the variable *t13* manually, you would count how many 1s there are, how many 2s, etc. Then, you would make a bar chart with either the number of observations or the percentage of observations on the y-axis for each value of *t13*. However, you don't need to waste your time doing this by hand.

Let a statistical analysis tool do it for you. You will need to learn how to use a statistical analysis tool to analyze data. I have used SAS, Excel, and

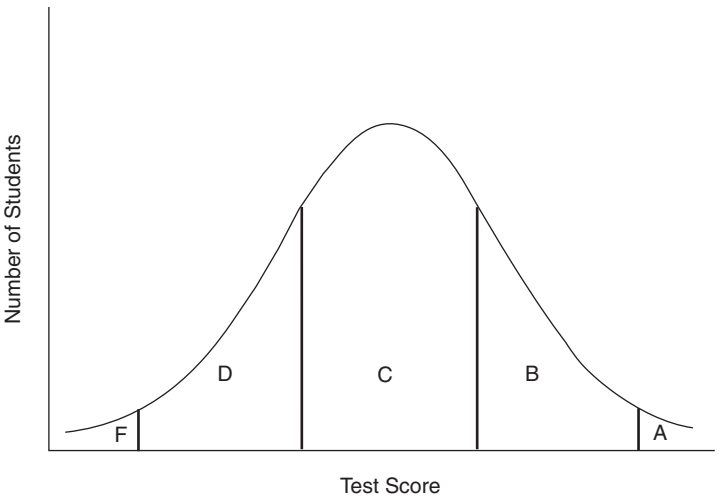

FIGURE 1.1
Example of a normal distribution

Stata in my career. My opinions regarding each are: SAS was fine when I worked for large organizations, but far too expensive when I had to pay for it myself. Excel is not powerful or straightforward enough for my purposes. Stata is relatively inexpensive (no yearly licensing fee), does everything I need, and is very easy to use (see *www.stata.com*). However, no matter which statistical software you use, the output should always look the same, and it is the interpretation of that output on which this book focuses.

Example The distributions of effort and size show that they are not normally distributed (Figures 1.2 and 1.3). The database contains few projects with a very high effort, or a very big size. It also contains many low effort, and small size projects. This is typical in a software development project database. Not only are the efforts and sizes not normally distributed in this sample, but we would not expect them to be normally distributed in the population of all software development projects.

To approximate a normal distribution, we must transform these variables. A common transformation is to take their natural log (*ln*). Taking the natural log makes large values smaller and brings the data closer together. For example, take two project sizes of 100 and 3000 function points. 3000 is much bigger than 100. If I take the *ln* of these numbers, I find that *ln*(100) = 4.6 and

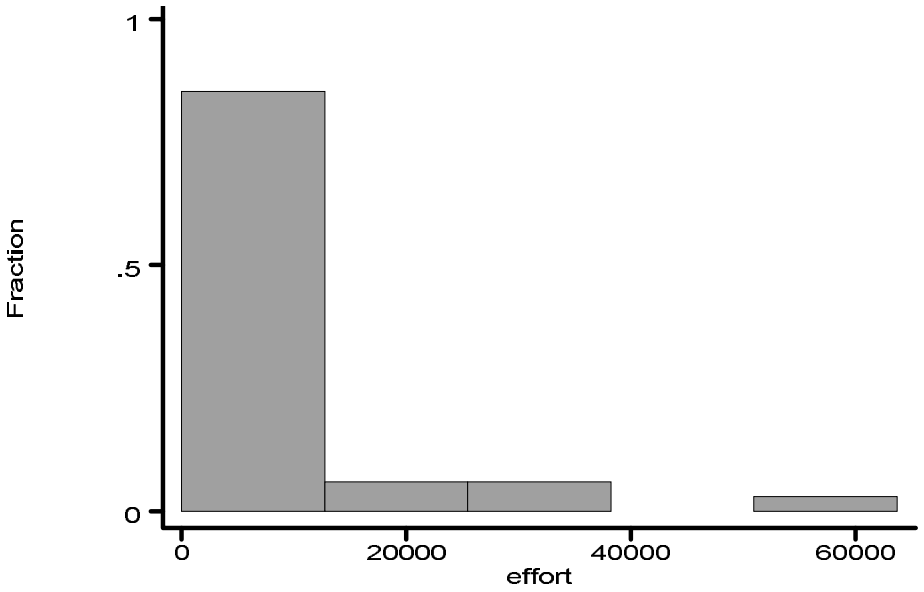

FIGURE 1.2
Distribution of *effort*

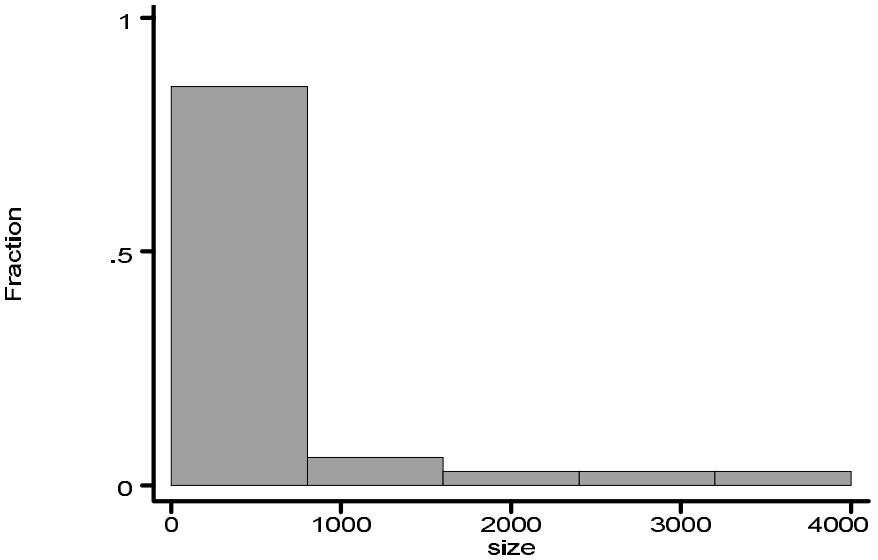

FIGURE 1.3
Distribution of *size*

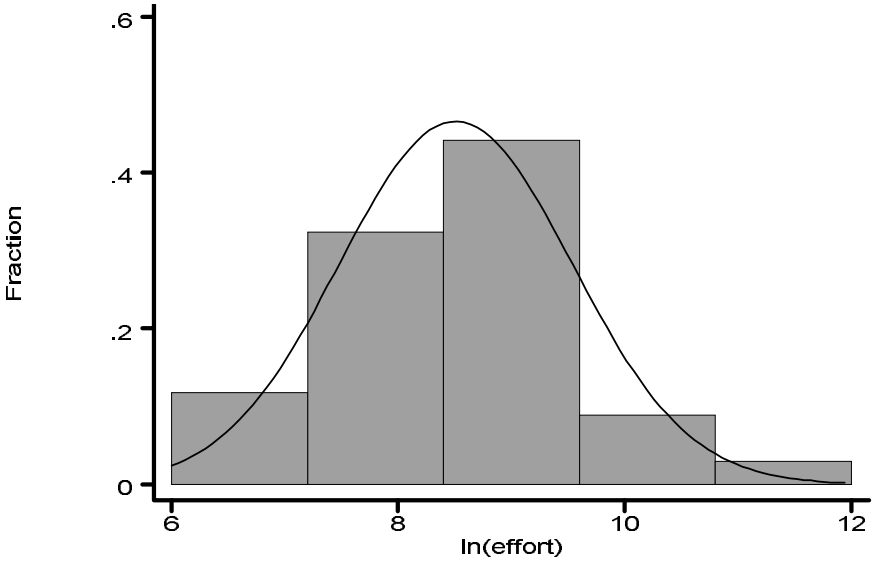

FIGURE 1.4
Distribution of *ln*(*effort*)

ln(3000) = 8.0. These transformed sizes are much closer together. As you can see, taking the natural log of effort and size more closely approximates a normal distribution (Figures 1.4 and 1.5).

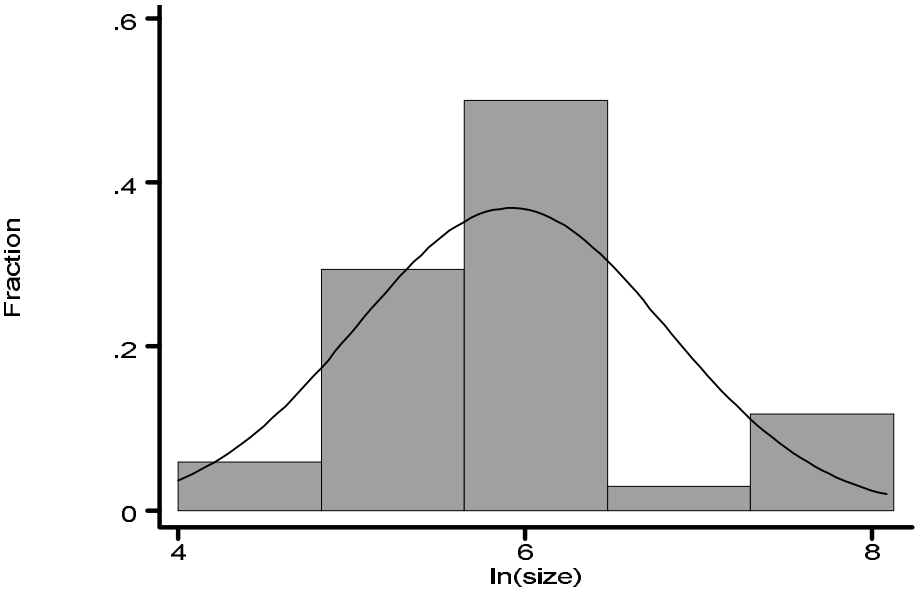

FIGURE 1.5
Distribution of *ln*(*size*)

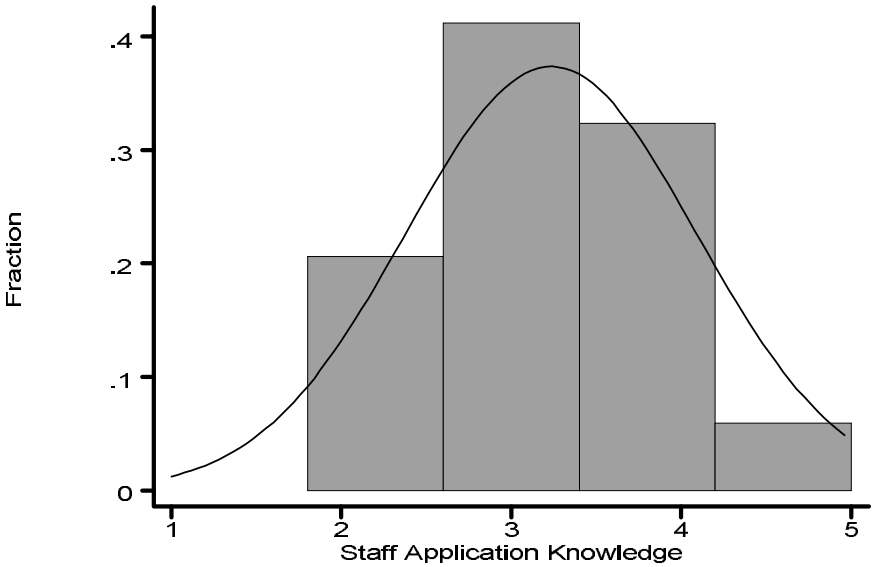

FIGURE 1.6
Distribution of *t13*

Graphs of staff application knowledge (*t13*) and staff tool skills (*t14*) look more normally distributed (Figures 1.6 and 1.7). Most projects have an average value of 3. Additionally, in the larger multi-company database from

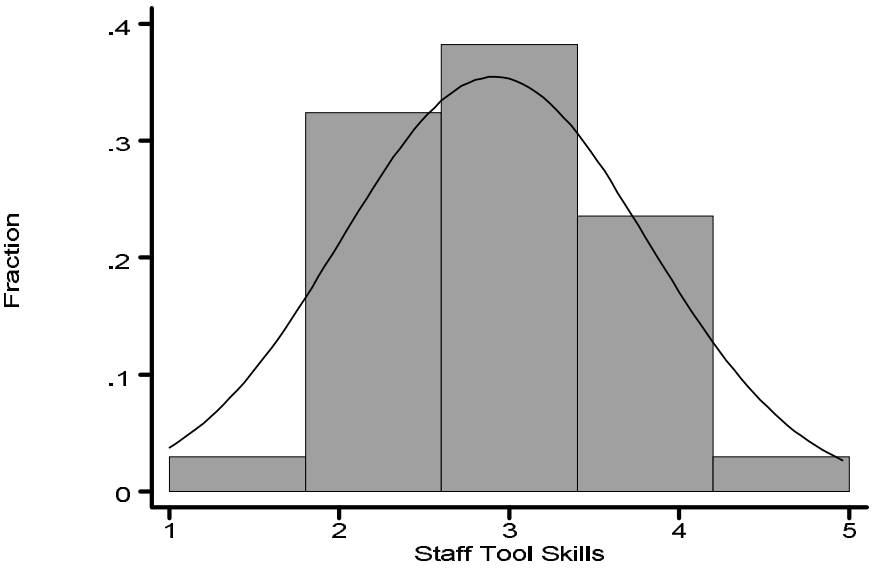

FIGURE 1.7
Distribution of *t14*

which this subset was taken, the distributions of these factors are approximately normal. In fact, the definitions of the factors were chosen especially so that most projects would be average. These variables do not need any transformation.

What to Watch Out For

- Just because the values of variables are numbers, it does not imply that they have any numerical sense. For example, application type (*app*) might have been coded as 1, 2, 3, and 4 instead of *CustServ*, *MIS*, *TransPro*, and *InfServ*. Application type (*app*) is a categorical variable with a nominal scale; that is, its values cannot be arranged in any meaningful order. I can arrange the values in any order I want: *MIS* before *CustServ*, for example. I suggest giving these types of variables meaningful names instead of numbers before you start analyzing the data. It will help you remember what they are. (You will learn more about variable types in Chapter 6.)
- On the other hand, there may be categorical variables with meaningful names that do have numerical sense. For example, staff application knowledge (*t13*) could have been coded as very low, low, average, high, and very high instead of 1, 2, 3, 4, and 5 (often referred to as a Likert scale). Staff application knowledge (*t13*) is a categorical variable whose

values can be arranged in a meaningful order. I suggest transforming these types of variables into ordered numbers before you start analyzing the data. Then, check to see if they are normally distributed. If they are approximately normally distributed, I treat them as numerical variables for the rest of the analysis. If they are not normally distributed, and I have no good reason to expect that in the population of all software development projects they would be, I treat them as categorical variables. It is common practice in the market research profession to treat Likert-type variables as numerical data. As you will see, it is easier to analyze numerical-type data than true categorical data.

Two-Dimensional Graphs I also make graphs of the dependent variable against each independent numerical variable. In this example, I am interested in the relationships between effort and size, effort and staff application knowledge (*t13*), and effort and staff tool skills (*t14*).

Why Do It? A picture is worth a thousand words. I highly recommend visualizing any relationship that might exist between the dependent and independent variables before running "blind" statistical tests. It is important to see if the relationship is linear as our statistical tests are based on linear relationships and will "ignore" non-linear relationships. A relationship is linear if you can fit one straight line through the data points, and this represents them well.

Example I plot these graphs using the transformed data. We can see in Figure 1.8 that there appears to be a linear relationship between *ln*(*effort*) and *ln*(*size*). As project size increases, the amount of effort needed increases. Figure 1.9 gives the impression that there is no relationship between effort and staff application knowledge (*t13*). Conversely, Figure 1.10 seems to suggest that less effort is required for projects with higher levels of staff tool skills (*t14*). These are first impressions that will be verified through statistical tests.

Another good reason to use a log transformation is to make a non-linear relationship more linear. Figure 1.11 shows the relationship between the variables *effort* and *size* before the log transformation. As you can see, the relationship in Figure 1.8 is much more linear than the relationship in Figure 1.11.

What to Watch Out For

- Non-linear relationships.
- Outliers—that is, any data points (projects) far away from the others. In an extreme case, an outlier can distort the scale, causing all the other

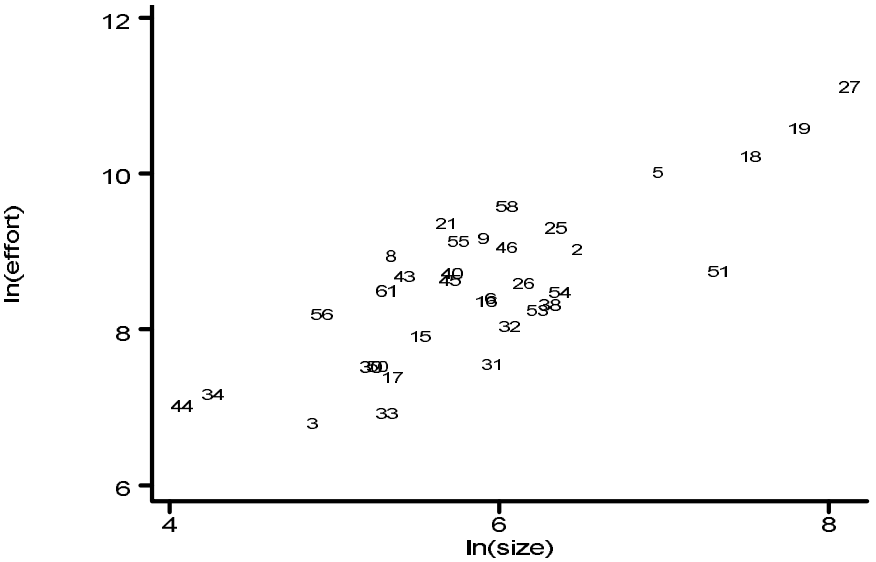

FIGURE 1.8
ln(*effort*) vs. *ln*(*size*)

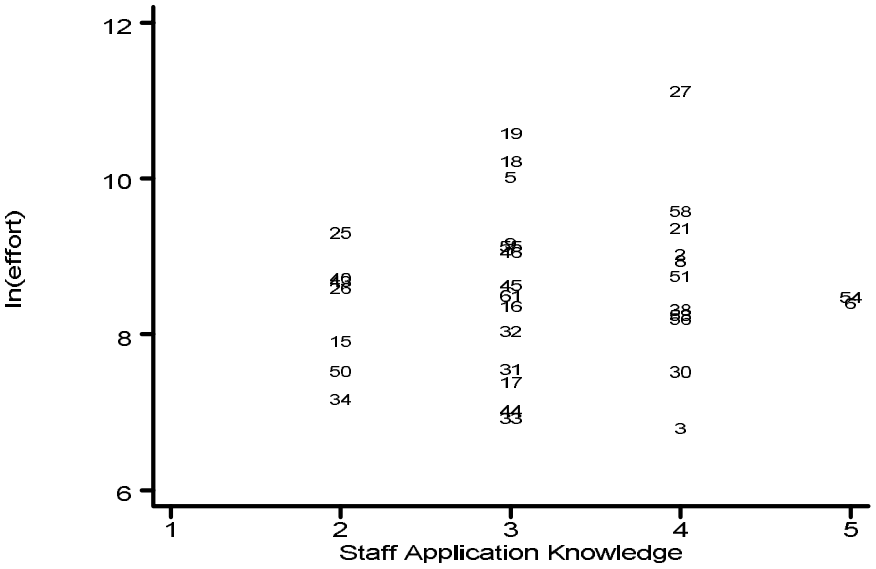

FIGURE 1.9
ln(*effort*) vs. *t13*

projects to look as if they are grouped together in a little cloud. All the straight lines fit to the data will try to go through the outlier, and will treat the cloud of data (that is, all the other projects) with less impor-

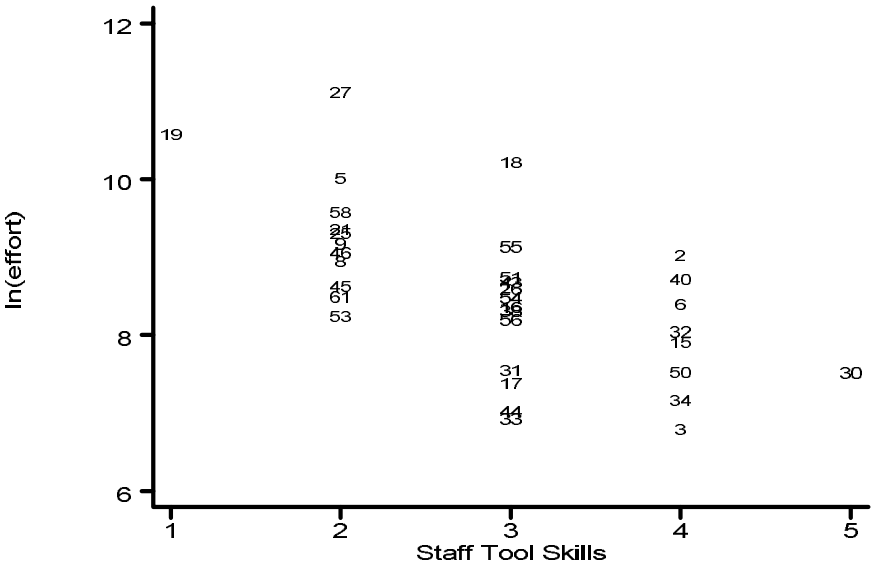

FIGURE 1.10
ln(*effort*) vs. *t14*

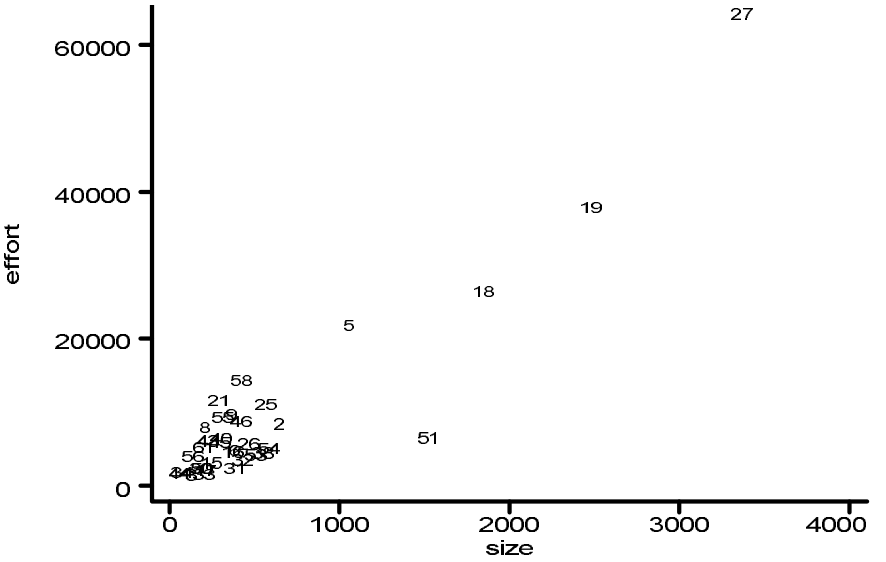

FIGURE 1.11
effort vs. *size*

tance. Remove the outlier(s) and re-plot the data to see if there is any relationship hidden in the cloud. See Chapter 2 for an example where an outlier is detected and removed.

Tables

I make tables of the average value of the dependent variable and the number of observations it is based on for each value of each categorical variable. In this example, the tables will show the average value of effort for each application type, and for Telon use.

Why Do It? We make tables to see if there is a big difference in the effort needed by category and to start formulating possible reasons for this.

Example From Example 1.4, we learn that on average, transaction processing (*TransPro*) applications require the highest effort, then customer service (*CustServ*) applications, then *MIS* applications, and finally, information service (*InfServ*) applications. Why is this? Answering this question will be important for the interpretation phase of the analysis. Example 1.5 tells us that, on average, projects that used Telon required almost twice as much effort as projects that did not. Is this because they were bigger in size, or could there be another explanation?

Example 1.4

```
. table app, c(n effort mean effort)

Application Type    N(effort)    mean(effort)
CustServ                 6            7872
MIS                      3            4434
TransPro                20           10816
InfServ                  5            4028
```

Example 1.5

```
. table telonuse, c(n effort mean effort)

  Telon Use    N(effort)    mean(effort)
  No              27            7497
  Yes              7           13510
```

What to Watch Out For Remember that we still need to check the relationships in Examples 1.4 and 1.5 to see if they are statistically significant.

Even if there appears to be a big difference in the average values, it may not really be true because one project with a high effort could have influenced a category's average.

Correlation Analysis

Another assumption of the statistical procedure I use to build a multi-variable model is that independent variables are independent; that is, they are not related to each other. In our example, there should be no strong relationships among the variables: *size*, *t13*, *t14*, *app*, and *telonuse*. There is a very quick way to check if the numerical variables, *size*, *t13*, and *t14*, are independent: correlation analysis. If some of the numerical variables were collected using an ordinal or quasi-interval Likert scale (like *t13* and *t14*), I use Spearman's rank correlation coefficient because it tests the relationships of orders rather than actual values. (See Chapter 6 for scale definitions.) Another important feature of Spearman's rank correlation coefficient is that it is less sensitive to extreme values than the standard Pearson correlation coefficient.

Two variables will be highly positively correlated if low ranked values of one are nearly always associated with low ranked values of the other, and high ranked values of one are nearly always associated with high ranked values of the other. For example, do projects with very low staff tool skills always have very low staff application knowledge, too; are average tool skills associated with average application knowledge, high tool skills with high application knowledge, etc.? If such a relationship is nearly always true, the correlation coefficient will be close to 1.

Two variables will be highly negatively correlated if low ranked values of one are nearly always associated with high ranked values of the other, and vice-versa. For example, do the smallest projects (smallest in size) always have the highest staff application knowledge, and do the biggest projects always have the lowest staff application knowledge? If such a situation is nearly always true, the correlation coefficient will be close to –1. Variables that are not correlated at all will have a correlation coefficient close to zero. You will learn more about correlation analysis in Chapter 6.

Why Do It? Perform a correlation analysis as a quick check to see if there are possible violations of the independence assumption. Later, as I build the multi-variable model, I will use this information. For the moment, I only make note of it.

Example Example 1.6 shows the statistical output for the Spearman's rank correlation coefficient test between the variables *size* and *t13*. The number of observations equals 34. The correlation coefficient is "*Spearman's rho*," which is 0.1952. Already it is clear that these two variables are not very correlated as this number is closer to 0 than 1. The "*Test of* H_o" tests if *size* and *t13* are independent (i.e., not correlated). If *Pr* > |*t*| = a number greater than 0.05, then *size* and *t13* are independent. Because 0.2686 > 0.05, we conclude that this is indeed the case. (*Pr* is an abbreviation for probability; *t* means that the t distribution was used to determine the probability. You will learn more about this in Chapter 6.)

Example 1.6

```
. spearman size t13

 Number of obs = 34
Spearman's rho = 0.1952
Test of Ho: size and t13 independent
      Pr > |t| = 0.2686
```

From Example 1.7, we learn that the variables *size* and *t14* have a Spearman's correlation coefficient of −0.3599. We cannot accept that *size* and *t14* are independent because 0.0365 is less than 0.05. Thus, we conclude that *size* and *t13* are negatively correlated.

Example 1.7

```
. spearman size t14

 Number of obs = 34
Spearman's rho = -0.3599
Test of Ho: size and t14 independent
      Pr > |t| =  0.0365
```

We conclude from the results in Example 1.8 that *t13* and *t14* are not correlated.

Example 1.8

```
. spearman t13 t14

 Number of obs = 34
Spearman's rho = -0.0898
Test of Ho: t13 and t14 independent
      Pr > |t| = 0.6134
```

What to Watch Out For

- If the absolute value of Spearman's rho is greater than or equal to 0.75, and the $Pr > |t|$ value equals 0.05 or less, then the two variables are strongly correlated and should not be included in the final model together.
- Many statistical analysis packages will automatically calculate the standard Pearson correlation coefficient unless you state otherwise. Make sure you request Spearman's correlation.
- It does not matter if you use the original variable (for example, *size*) or the transformed variable (*ln*(*size*)) to calculate Spearman's correlation; the results will be the same.

Categorical Variable Tests Now that we have checked the numerical variables' independence with correlation analysis, perhaps you are asking: What about the categorical variables? It takes much more time to check the independence of every possible relationship between the categorical variables and between the categorical variables and numerical variables, especially in a large database. It is for this reason that I only carry out these checks on the independent variables present in the final multi-variable model in Step 5, when I check the model.

Stepwise Regression Analysis

Performing multiple regression analyses allows us to determine the relative importance of each independent, numerical variable's relationship (*ln*(*size*), *t13*, *t14*) to the dependent variable (*ln*(*effort*)).

Why Do It? Because stepwise regression analysis is automatic and very simple to run, I always like to see how good of a model can be built just with the numerical data. In addition to learning if the non-categorical variables collected are very important indicators of effort, this also gives me a quick idea of what performance the categorical data is going to have to beat.

Example The output in Example 1.9 shows the results of running a forward stepwise regression procedure on our data set. Forward stepwise regression means that the model starts "empty" and then the variables most related to *leffort* (abbreviation of *ln*(*effort*) in statistical output) are added one by one in order of importance until no other variable can be added to improve the model. You must run this procedure using the transformed variables.

You can see that first, *lsize* (abbreviation of *ln*(*size*) in statistical output) is added, then *t14* is added. No further variation in *leffort* is explained by *t13*, so it is left out of the model. In Chapter 6, you will learn how to interpret every part of this output; for now, I will just concentrate on the values in bold. These are the values that I look at to determine the performance and significance of the model. I look at the number of observations (*Number of obs*) to see if the model was built using all the projects. The model was built using all 34 observations. I look at *Prob > F* to determine if the model is significant, in other words, can I believe this model? (*Prob* is an abbreviation for probability; *F* means that the F distribution was used to determine the probability. You will learn more about this in Chapter 6.) If *Prob > F* is a number less than or equal to 0.05, then I accept the model. Here it is 0, so the model is significant. I look at the adjusted R-squared value (*Adj R-squared*) to determine the performance of the model. The closer it is to 1, the better. The *Adj R-squared* of 0.7288 means that this model explains nearly 73% (72.88%) of the variation in *leffort*. This is a very good result. This means that even without the categorical variables, I am sure to come up with a model than explains 73% of the variation in effort. I am very interested in finding out more about which variables explain this variation.

I can see from the output that *lsize* and *t14* are the RHS explanatory variables. I also check the significance of each explanatory variable and the constant (*_cons*) in the column *P > |t|*. If *P > |t|* is a number less than or equal to 0.05, then the individual variable is significant; that is, it is not in the model by chance. (*P* is yet another abbreviation for probability; *t* means that the t distribution was used to determine the probability.)

Example 1.9

```
. sw regress leffort lsize t13 t14, pe(.05)
begin with empty model

             p = 0.0000 < 0.0500   adding   lsize
             p = 0.0019 < 0.0500   adding   t14

Source         SS       df       MS                  Number of obs =      34
Model     25.9802069     2   12.9901035               F(2,31)       =   45.35
Residual  8.88042769    31   .286465409               Prob > F      =  0.0000
Total     34.8606346    33   1.05638287               R-squared     =  0.7453
                                                      Adj R-squared =  0.7288
                                                      Root MSE      =  .53522

leffort      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
lsize       .7678266   .1148813     6.684   0.000     .5335247    1.002129
t14        -.3856721   .1138331    -3.388   0.002    -.6178361    -.153508
_cons       5.088876   .8764331     5.806   0.000     3.301379    6.876373
```

The output in Example 1.10 shows the results of running a backward stepwise regression procedure on our data set. Backward stepwise regression means that the model starts "full" (with all the variables) and then the variables least related to effort are removed one by one in order of unimportance until no further variable can be removed to improve the model. You can see here that *t13* was removed from the model. In this case, the results are the same for forward and backward stepwise regression; however, this is not always the case. Things get more complicated when some variables have missing observations.

Example 1.10

```
. sw regress leffort 1size t13 t14, pr(.05)
begin with full model
p = 0.6280 >= 0.0500  removing t13

Source       SS       df       MS              Number of obs =      34
Model     25.9802069   2  12.9901035           F(2,31)       =   45.35
Residual  8.88042769   31  .286465409          Prob > F      =  0.0000
Total     34.8606346   33  1.05638287          R-squared     =  0.7453
                                               Adj R-squared =  0.7288
                                               Root MSE      =  .53522

leffort      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
lsize       .7678266   .1148813     6.684   0.000     .5335247   1.002129
t14        -.3856721   .1138331    -3.388   0.002    -.6178361   -.153508
_cons       5.088876   .8764331     5.806   0.000     3.301379   6.876373
```

What to Watch Out For Watch out for variables with lots of missing values. The stepwise regression procedure only takes into account observations with non-missing values for **all** variables specified. For example, if *t13* is missing for half the projects, then half the projects will not be used. Check the number of observations used in the model. You may keep coming up with models that explain a large amount of the variation for a small amount of the data. If this happens, run the stepwise procedures using only the variables available for nearly every project.

Building the Multi-Variable Model

I call the technique I've developed to build the multi-variable model "stepwise ANOVA" (analysis of variance). It is very similar to forward stepwise regression except I use an analysis of variance procedure to build models

with categorical variables. You will learn more about analysis of variance in Chapter 6. For the moment, you just need to know that this procedure allows us to determine the influence of numerical **and** categorical variables on the dependent variable, *leffort*. The model starts "empty" and then the variables most related to *leffort* are added one by one in order of importance until no other variable can be added to improve the model. The procedure is very labor-intensive because I make the decisions at each step myself; it is not automatically done by the computer. Although I am sure this could be automated, there are some advantages to doing it yourself. As you carry out the steps, you will develop a better understanding of the data. In addition, in the real world, a database often contains many missing values and it is not always clear which variable should be added at each step. Sometimes you need to follow more than one path to find the best model. In the following example, I will show you the simplest case using our 34-project, 6-variable database with no missing values. My goal for this chapter is that you understand the methodology. The four case studies in Chapters 2 through 5 present more complicated analyses, and will focus on interpreting the output.

Example

Determine Best One-Variable Model First, we want to find the best one-variable model. Which variable, *lsize*, *t13*, *t14*, *app*, or *telonuse*, explains the most variation in *leffort*? I run regression procedures for the numerical variables and ANOVA procedures for the categorical variables to determine this. In practice, I do not print all the output. I save it in output listing files and record by hand the key information in a summary sheet. Sidebar 1.2 shows a typical summary sheet. I note the date that I carried out the analysis, the directory where I saved the files, and the names of the data file, the procedure file(s), and the output file(s). I may want to look at them again in the future, and if I don't note their names now, I may never find them again! We are going to be creating lots of procedures and generating lots of output, so it is important to be organized. I also note the name of the dependent variable.

Now I am ready to look at the output file and record the performance of the models. In the summary sheet, I record data only for significant variables. For the regression models, a variable is highly significant if its $P > |t|$ value is 0.05 or less. In this case, I do not record the actual value; I just note the number of observations, the variable's effect on effort, and the adjusted R-squared value. If the significance is borderline, that is, if $P > |t|$ is a number between 0.05 and 0.10, I note its value. If the constant is not significant, I note it in the *Comments* column. If you are analyzing a very small database, you might like to record these values for every variable—significant or not.

Personally, I have found that it is not worth the effort for databases with many variables. If I need this information later, I can easily go back and look at the output file.

For the ANOVA models, I do the same except I look at a variable's *Prob > F* value to determine if the variable is significant. The effect of a categorical variable depends on the different types. For example, using Telon (*telonuse* = Yes) will have one effect on *leffort* and not using Telon (*telonuse* = No) will have a different effect on *leffort*. You cannot determine the effect from the ANOVA table.

In Example 1.11, I have highlighted the key numbers in bold. I see that there is a very significant relationship between *leffort* and *lsize* ($P > |t| = 0.000$): *lsize* explains 64% of the variation in *leffort*. The coefficient of *lsize* (*Coef.*) is a positive number (0.9298). This means that *leffort* increases with increasing *lsize*. The model was fit using data from 34 projects. I add this information to the summary sheet (Sidebar 1.2).

Example 1.11

```
. regress leffort 1size

      Source |       SS       df       MS              Number of obs =      34
-------------+------------------------------           F(1,32)       =   59.67
       Model |  22.6919055     1  22.6919055           Prob > F      =  0.0000
    Residual |  12.1687291    32  .380272786           R-squared     =  0.6509
-------------+------------------------------           Adj R-squared =  0.6400
       Total |  34.8606346    33  1.05638287           Root MSE      =  .61666

------------------------------------------------------------------------------
     leffort |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       lsize |   .9297666   .1203611     7.725   0.000     .6845991    1.174934
       _cons |   3.007431   .7201766     4.176   0.000      1.54048    4.474383
------------------------------------------------------------------------------
```

In Example 1.12, I see that there is not a significant relationship between *leffort* and *t13*. Therefore, I do not even look at the coefficient of *t13*. I note nothing and move on to the next model.

Example 1.12

```
. regress leffort t13

      Source |       SS       df       MS              Number of obs =      34
-------------+------------------------------           F(1,32)       =    0.39
       Model |  .421933391     1  .421933391           Prob > F      =  0.5357
    Residual |  34.4387012    32  1.07620941           R-squared     =  0.0121
-------------+------------------------------           Adj R-squared =  0.0188
       Total |  34.8606346    33  1.05638287           Root MSE      = -1.0374
```

leffort	Coef.	Std. Err.	t	**P>|t|**	[95% Conf.	Interval]
t13	.1322679	.2112423	0.626	**0.536**	-.2980186	.5625544
_cons	8.082423	.706209	11.445	0.000	6.643923	9.520924

In Example 1.13, I see that there is a very significant relationship between *leffort* and *t14* : *t14* explains 36% of the variation in *leffort*. The coefficient of *t14* is negative. This means that *leffort* decreases with increasing *t14*. The model was fit using data from 34 projects. I add this information to the summary sheet (Sidebar 1.2).

Example 1.13

```
. regress leffort t14
```

Source	SS	df	MS		
Model	13.1834553	1	13.1834553	**Number of obs**	**= 34**
Residual	21.6771793	32	.677411853	F(1,32)	= 19.46
Total	34.8606346	33	1.05638287	Prob > F	= 0.0001
				R-squared	= 0.3782
				Adj R-squared	**= 0.3587**
				Root MSE	= .82305

leffort	**Coef.**	Std. Err.	t	**P>|t|**	[95% Conf.	Interval]
t14	**-.7022183**	.1591783	-4.412	**0.000**	-1.026454	-.3779827
_cons	10.55504	.4845066	21.785	**0.000**	9.568136	11.54195

In Example 1.14, I see that there is no significant relationship between *leffort* and *app*. I note nothing and move on to the next model.

Example 1.14

```
. anova leffort app
```

Number of obs	= 34	R-squared	= 0.0210
Root MSE	= 1.06659	Adj R-squared	= -0.0769

Source	Partial SS	df	MS	F	**Prob > F**
Model	.732134098	3	.244044699	0.21	0.8855
app	.732134098	3	.244044699	0.21	**0.8855**
Residual	34.1285005	30	1.13761668		
Total	34.8606346	33	1.05638287		

In Example 1.15, I see that there is a borderline significant relationship between *leffort* and *telonuse* : *telonuse* explains 8% of the variation in *leffort*. The model was fit using data from 34 projects. I add this information to the summary sheet (Sidebar 1.2).

Example 1.15

```
. anova leffort telonuse

Number of obs = 34              R-squared = 0.1094
     Root MSE = .984978    Adj R-squared = 0.0816

Source      Partial SS  df       MS        F     Prob > F
Model       3.81479355   1   3.81479355   3.93    0.0560
telonuse    3.81479355   1   3.81479355   3.93    0.0560
Residual    31.0458411  32   .970182533
Total       34.8606346  33   1.05638287
```

SIDEBAR 1.2
STATISTICAL OUTPUT SUMMARY SHEET

Date: 01/03/2001
Directory: C:\my documents\data analysis book\example34\
Data File: bankdata34.dta
Procedure Files: *var.do (* = one, two, three, etc.)
Output Files: *var.log
Dependent Variable: *leffort*

Variables	Num Obs	Effect	Adj R^2	Significance of Added Variable	Comments
1-variable models					
lsize	**34**	+	**0.64**		
t14	34	–	0.36		
telonuse	34		0.08	.056	
2-variable models					
with *lsize*					
t14	**34**	**–**	**0.73**		**best model, sign. = 0.0000**
3-variable models					
with *lsize, t14*					
none significant					no further improvement possible

Once I have recorded all of the output in the summary sheet, I select the variable that explains the most variation in *leffort*. In this step, it is obviously *lsize*. There is no doubt about it. Then I ask myself: Does the relationship between *leffort* and *lsize* make sense? Does it correspond to the graph of *leffort* as a function of *lsize* (Figure 1.8)? Yes, it does, so I add *lsize* to the model and continue with the next step.

Determine Best Two-Variable Model Next I want to determine which variable, *t13*, *t14*, *app*, or *telonuse*, in addition to *lsize*, explains the most variation in *leffort*. I add *lsize* to my regression and ANOVA procedures and run them again. I record the output in the same summary sheet (Sidebar1.2). What I am principally concerned with at this stage is if the additional variable is significant. So first I look at $P > |t|$ value of this variable. If it is not significant, I record nothing and move on to the next model.

In Example 1.16, I see that *t13* is not significant (0.595).

Example 1.16

```
. regress leffort lsize t13

Source       SS       df       MS              Number of obs =      34
Model     22.8042808   2   11.4021404           F(2,31)       =   29.32
Residual  12.0563538   31  .388914638           Prob > F      =  0.0000
Total     34.8606346   33  1.05638287           R-squared     =  0.6542
                                                Adj R-squared =  0.6318
                                                Root MSE      =  .62363

leffort     Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
lsize        .943487   .1243685    7.586   0.000     .6898359    1.197138
t13         -.0697449  .1297491   -0.538   0.595      -.33437    .1948801
_cons        3.151871  .7763016    4.060   0.000     1.568593    4.735149
```

In Example 1.17, I learn that *t14* is significant (0.002): *lsize* and *t14* together explain 73% of the variation in *leffort*. The coefficient of *t14* is a negative number. This means that *leffort* decreases with increasing *t14*. This is the same effect that we found in the one-variable model. If the effect was different in this model, that could signal something strange going on between *lsize* and *t13*, and I would look into their relationship more closely. *lsize* and the constant (*_cons*) are still significant. If they were not, I would note this in the *Comments* column. Again, this model was built using data from 34 projects.

Example 1.17

```
. regress leffort lsize t14

Source       SS       df       MS              Number of obs =      34
Model     25.9802069   2   12.9901035           F(2,31)       =   45.35
Residual  8.88042769   31  .286465409           Prob > F      =  0.0000
Total     34.8606346   33  1.05638287           R-squared     =  0.7453
                                                Adj R-squared =  0.7288
                                                Root MSE      =  .53522
```

leffort	**Coef.**	Std. Err.	t	**P>|t|**	[95% Conf.	Interval]
lsize	.7678266	.1148813	6.684	**0.000**	.5335247	1.002129
t14	**-.3856721**	.1138331	-3.388	**0.002**	-.6178361	-.153508
_cons	5.088876	.8764331	5.806	**0.000**	3.301379	6.876373

In Examples 1.18 and 1.19, I see that *app* and *telonuse* are not significant (0.6938 and 0.8876).

Example 1.18

```
. anova leffort lsize app, category (app)
```

Number of obs =	34	R-squared	= 0.6677		
Root MSE	= .63204	Adj R-squared	= 0.6218		

Source	Partial SS	df	MS	F	**Prob > F**
Model	23.2758606	4	5.81896516	14.57	0.0000
lsize	22.5437265	1	22.5437265	56.43	0.0000
app	.583955179	3	.194651726	0.49	**0.6938**
Residual	11.584774	29	.399474964		
Total	34.8606346	33	1.05638287		

Example 1.19

```
. anova leffort lsize telonuse, category (telonuse)
```

Number of obs =	34	R-squared	= 0.6512		
Root MSE	= .626325	Adj R-squared	= 0.6287		

Source	Partial SS	df	MS	F	**Prob > F**
Model	22.6998727	2	11.3499363	28.93	0.0000
lsize	18.8850791	1	18.8850791	48.14	0.0000
telonuse	.007967193	1	.007967193	0.02	**0.8876**
Residual	12.1607619	31	.392282644		
Total	34.8606346	33	1.05638287		

Again, the decision is obvious: The best two-variable model of *leffort* is *lsize* and *t14*. Does the relationship between *t14* and *leffort* make sense? Does it correspond to the graph of *leffort* as a function of *t14*? If yes, then we can build on this model.

Determine Best Three-Variable Model Next I want to determine which variable, *t13*, *app*, or *telonuse*, in addition to *lsize* and *t14*, explains the most variation in *leffort*. I add *t14* to my regression and ANOVA procedures from the previous step and run them again. I record the output in the same summary sheet (Sidebar 1.2). As in the previous step, what I am principally concerned with at this stage is if the additional variable is significant. If it is

not significant, I record nothing and move on to the next model. Let's look at the models (Examples 1.20, 1.21, and 1.22).

Example 1.20

```
. regress leffort lsize t14 t13

      Source |       SS       df       MS              Number of obs =      34
-------------+------------------------------           F(3, 30)      =   29.57
       Model |  26.0505804     3  8.68352679            Prob > F      =  0.0000
    Residual |  8.81005423    30  .293668474           R-squared     =  0.7473
-------------+------------------------------           Adj R-squared =  0.7220
       Total |  34.8606346    33  1.05638287           Root MSE      =  .54191

------------------------------------------------------------------------------
     leffort |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       lsize |   .7796095    .118781     6.563   0.000     .5370263    1.022193
         t14 |   -.383488    .1153417   -3.325   0.002    -.6190471   -.1479289
         t13 |   -.055234    .1128317   -0.490   0.628     -.285667    .175199
       _cons |   5.191477    .9117996    5.694   0.000     3.329334    7.05362
------------------------------------------------------------------------------
```

Example 1.21

```
. anova leffort lsize t14 app, category (app)

                   Number of obs =      34    R-squared     =  0.7478
                   Root MSE      =  .560325   Adj R-squared =  0.7028

                          Partial SS   df         MS          F     Prob > F
-----------------------------------------------------------------------------
Model                     26.0696499    5  5.21392998      16.61     0.0000
lsize                     12.3571403    1  12.3571403      39.36     0.0000
t14                       2.79378926    1  2.79378926       8.90     0.0059
app                       .089442988    3  .029814329       0.09     0.9622
Residual                   8.7909847   28  .313963739
-----------------------------------------------------------------------------
Total                     34.8606346   33  1.05638287
```

Example 1.22

```
. anova leffort lsize t14 telonuse, category(telonuse)

                   Number of obs =      34    R-squared     =  0.7487
                   Root MSE      =  .540403   Adj R-squared =  0.7236

                          Partial SS   df         MS          F     Prob > F
-----------------------------------------------------------------------------
Model                      26.099584    3  8.69986134      29.79     0.0000
lsize                      12.434034    1   12.434034      42.58     0.0000
t14                       3.39971135    1  3.39971135      11.64     0.0019
telonuse                  .119377093    1  .119377093       0.41     0.5274
Residual                   8.7610506   30   .29203502
-----------------------------------------------------------------------------
Total                     34.8606346   33  1.05638287
```

None of the additional variables in the three models (Examples 1.20, 1.21, and 1.22) are significant.

The Final Model The stepwise ANOVA procedure ends as no further improvement in the model is possible. The best model is the two-variable model: *leffort* as a function of *lsize* and *t14*. No categorical variables were significant in this example, so this model is the same model found by the automatic stepwise regression procedure. I check one final time that the relationships in the final model (Example 1.23) make sense. We see that *lsize* has a positive coefficient. This means that the bigger the application size, the greater the development effort required. Yes, this makes sense to me. I would expect bigger projects to require more effort. The coefficient of *t14*, staff tool skills, is negative. This means that effort decreases with increasing staff tool skills. Projects with very high staff tool skills required less effort than projects with very low staff tool skills, everything else being constant. Yes, this makes sense to me, too. Print the final model's output and save it.

Example 1.23

```
. regress leffort lsize t14
```

Source	SS	df	MS		
Model	25.9802069	2	12.9901035	Number of obs	= 34
Residual	8.88042769	31	.286465409	F(2, 31)	= 45.35
Total	34.8606346	33	1.05638287	**Prob > F**	**= 0.0000**
				R-squared	= 0.7453
				Adj R-squared	= 0.7288
				Root MSE	= .53522

leffort	**Coef.**	Std. Err.	t	P>|t|	[95% Conf.	Interval]
lsize	**.7678266**	.1148813	6.684	0.000	.5335247	1.002129
t14	**-.3856721**	.1138331	-3.388	0.002	-.6178361	-.153508
_cons	**5.088876**	.8764331	5.806	0.000	3.301379	6.876373

On the summary sheet, I note the significance of the final model. This is the *Prob > F* value at the top of the output. The model is significant at the 0.0000 level. This is Stata's way of indicating a number smaller than 0.00005. This means that there is less than a 0.005% chance that all the variables in the model (*lsize* and *t14*) are not related to *leffort*. (More information about how to interpret regression output can be found in Chapter 6.)

What to Watch Out For

- Be sure to use an ANOVA procedure that analyzes the variance of unbalanced data sets, that is, data sets that do not contain the same number of observations for each categorical value. I have yet to see a

"balanced" software development database. In Stata, the procedure is called "ANOVA."

- Use the transformed variables in the model.
- Some models may contain variables with lots of missing values. It might be better to build on the second best model if it is based on more projects.
- If the decision is not obvious, follow more than one path to its final model (see Chapters 4 and 5). You will end up with more than one final model.
- Always ask yourself at each step if a model makes sense. If a model does not make sense, use the next best model.

Checking the Model

Before we can accept the final model found in the previous step, we must check that the assumptions underlying the statistical tests used have not been violated. In particular, this means checking that:

- Independent numerical variables are approximately normally distributed. (We did this in the preliminary analyses.)
- Independent variables are not strongly related to each other. (We did this partially during the preliminary analyses; now, we need to complete it.)
- The errors in our model should be random and normally distributed. (We still need to do this.)

In addition, we also need to check that no single project or small number of projects has an overly strong influence on the results.

Numerical Variable Checks

We already calculated the correlation coefficients of numerical variables in our preliminary analyses and noted them. Now that I have my final model, I need to check that all the independent numerical variables present in the final model are not strongly linearly related to each other. In other words, I need to check for multicollinearity problems. Why would this cause a problem? If two or more explanatory variables are very highly correlated, it is sometimes not possible for the statistical analysis software to separate their independent effects and you will end up with some strange results. Exactly when this will happen is not predictable. So, it is up to you to check the correlations between all numerical variables. Because my model only depends on *lsize* and *t14*, I just

need to check their correlation with each other. To avoid multicollinearity problems, I do not allow any two variables with an absolute value of Spearman's rho greater than or equal to 0.75 in the final model together. From our preliminary correlation analysis, we learned that *size*[1] and *t14* are slightly negatively correlated; they have a significant Spearman's correlation coefficient of –0.3599. Thus, there are no multicollinearity problems with this model.

You should also be aware that there is always the possibility that a variable outside the analysis is really influencing the results. For example, let's say I have two variables, my weight and the outdoor temperature. I find that my weight increases when it is hot and decreases when it is cold. I develop a model that shows my weight as a function of outdoor temperature. If I did not use my common sense, I could even conclude that the high outdoor temperature causes my weight gain. However, there is an important variable that I did not collect which is the real cause of any weight gain or loss—my ice cream consumption. When it is hot outside, I eat more ice cream, and when it is cold, I eat much less. My ice cream consumption and the outdoor temperature are therefore highly correlated. The model should really be my weight as a function of my ice cream consumption. This model is also more useful because my ice cream consumption is within my control, whereas the outdoor temperature is not. In this case, the outdoor temperature is confounded[2] with my ice cream consumption and the only way to detect this is to think about the results. Always ask yourself if your results make sense and if there could be any other explanation for them. Unfortunately, we are less likely to ask questions and more likely to believe a result when it proves our point.

Categorical Variable Checks

Strongly related categorical variables can cause problems similar to those caused by numerical variables. Unfortunately, strong relationships involving categorical variables are much more difficult to detect. We do not have any categorical variables in our final effort model, so we do not need to do these checks for our example. However, **if** we had found that *telonuse* and *app* were both in the model, how would we check that they are not related to each other or to the numerical variables in the model?

To determine if there is a relationship between a categorical variable and a numerical variable, I use an analysis of variance procedure. Let's take *app* and *t14* in Example 1.24. Does *app* explain any variance in *t14*?

1. Remember that the Spearman's correlation coefficients of *size* and *t14*, and *lsize* and *t14*, are identical.
2. Confound means to mistake one variable's effect for another's.

Example 1.24

```
. anova t14 app

                  Number of obs =     34    R-squared     = 0.1023
                  Root MSE      = .894427   Adj R-squared = 0.0125

Source      Partial SS   df       MS          F      Prob > F
Model       2.73529412    3   .911764706     1.14     0.3489
app         2.73529412    3   .911764706     1.14     0.3489
Residual         24.00   30          .80
Total       26.7352941   33   .810160428
```

Example 1.24 shows that there is no significant relationship between *app* and *t14* (the *Prob > F* value for *app* is a number greater than 0.05). I run ANOVA procedures for every categorical/numerical variable combination in the final model. (Note that the numerical variable must be the dependent LHS variable.) If I find a very strong relationship, I will not include the two variables together in the same model. I define "very strong relationship" as one variable explaining more than 75% of the variation in another.

I would like to point out here that we can get a pretty good idea about which variables are related to each other just by looking at the list of variables that are significant at each step as we build the one-variable, two-variable, three-variable, etc. models. In the statistical output sheet, Sidebar 1.2, we see that *telonuse* is an important variable in the one-variable model. However, once *lsize* has been added to the model, *telonuse* does not appear in the two-variable model. This means that there is probably a relationship between *telonuse* and *lsize*. Let's check (Example 1.25):

Example 1.25

```
. anova lsize telonuse

                  Number of obs =     34    R-squared     = 0.1543
                  Root MSE      = .832914   Adj R-squared = 0.1279

Source      Partial SS   df       MS          F      Prob > F
Model       4.04976176    1   4.04976176     5.84     0.0216
telonuse    4.04976176    1   4.04976176     5.84     0.0216
Residual    22.1998613   32   .693745665
Total       26.2496231   33   .795443123
```

Yes, there is a significant relationship between *lsize* and *telonuse*. The use of Telon explains about 13% of the variance in *lsize*. Example 1.26 shows that applications that used Telon were much bigger than applications that

did not. So, the larger effort required by applications that used Telon (Example 1.5) may not be due to Telon use per se, but because the applications were bigger. Once size has been added to the effort model, Telon use is no longer important; size is a much more important driver of effort. I learn as I analyze. Had this all been done automatically, I may not have noticed this relationship.

Example 1.26

```
. table telonuse, c(mean size)

Telon Use       mean(size)
No                     455
Yes                   1056
```

It is more difficult to determine if there is an important relationship between two categorical variables. To check this, I first calculate the chi-square statistic to test for independence. From this I learn if there is a significant relationship between two categorical variables, but not the extent of the relationship. (You will learn more about the chi-square test in Chapter 6.) In Example 1.27, I am interested in the *Pr* value (in bold). *Pr* is the probability that we are making a mistake if we say that there is a relationship between two variables. If the value of *Pr* is less than or equal to 0.05, we can accept that there is a relationship between the two variables. Here, *Pr* = 0.069, so I conclude that there is no significant relationship between the two variables.

Example 1.27

```
. tabulate app telonuse, chi2

Application Type      Telon Use
                        No      Yes      Total
CustServ                 6        0          6
MIS                      3        0          3
TransPro                16        4         20
InfServ                  2        3          5
Total                   27        7         34

      Pearson chi2(3) = 7.0878    Pr = 0.069
```

If there is a significant relationship, I need to look closely at the two variables and judge for myself if they are so strongly related that there could be a problem. For example, if application type (*app*) and Telon use (*telonuse*) had been significantly related, I would first look closely at Example 1.27. There I would learn that no customer service (*CustServ*) or *MIS* application used

Telon. Of the seven projects that used Telon, there is a split between transaction processing (*TransPro*) applications (a high-effort category; see Example 1.4) and information service (*InfServ*) applications (a low-effort category). Thus, the high effort for Telon use (see Example 1.5) is not due to an overrepresentation of high-effort transaction processing applications. In fact, the majority of projects that did not use Telon are transaction processing applications. I conclude that any relationship between Telon use and effort cannot be explained by the relationship between application type and Telon use; i.e. application type and Telon use are not confounded.

If I find any problems in the final model, I return to the step where I added the correlated/confounded variable to the variables already present in the model, take the second best choice, and rebuild the model from there. I do not carry out any further checks. The model is not valid, so there is no point. We have to start again. (See Chapter 5 for an example of confounded categorical variables.)

Testing the Residuals

In a well-fitted model, there should be no pattern to the errors (residuals) plotted against the fitted values. The term "fitted value" refers to the *leffort* predicted by our model; the term "residual" is used to express the difference between the actual *leffort* and the predicted *leffort* for each project. Your statistical analysis tool should calculate the predicted values and residuals automatically for you. The errors of our model should be random. For example, we should not be consistently overestimating small efforts and underestimating large efforts. It is always a good idea to plot this relationship and take a look. If you see a pattern, it means that there is a problem with your model. If there is a problem with the final model, then try the second best model. If there is a problem with the second best model, then try the third best model, and so on. In Figure 1.12, I see no pattern in the residuals of our final model.

In addition, the residuals should be normally distributed. We can see in Figure 1.13 that they are approximately normally distributed. You will learn more about residuals in Chapter 6.

Detecting Influential Observations

How much is our final model affected by any one project or subset of our data? If we dropped one project from our database, would our model be completely different? I certainly hope not. But we can do better than hope; we can check the model's sensitivity to individual observations. Projects

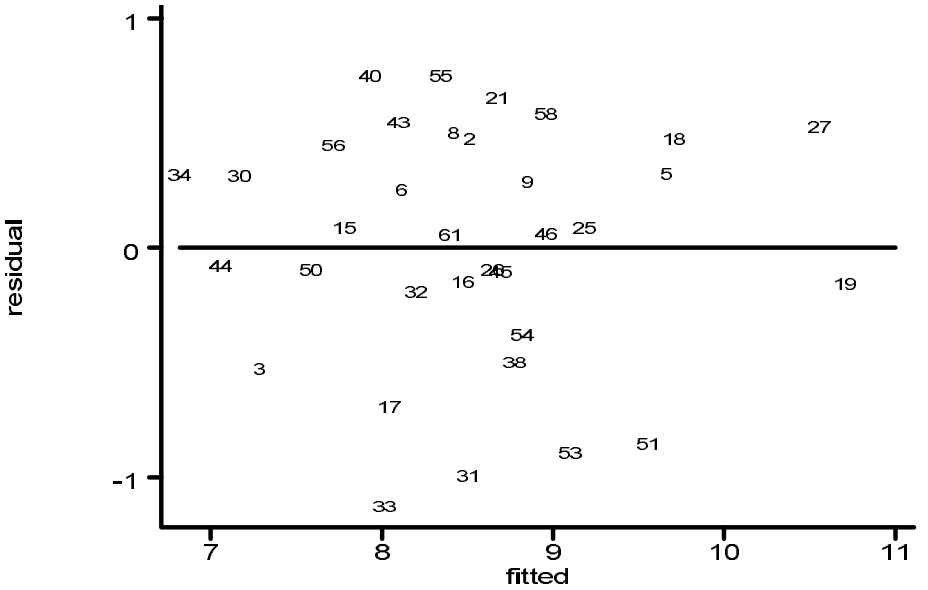

FIGURE 1.12
Residuals vs. fitted values

with large predicted errors (residuals) and/or projects very different from other project's values for at least one of the independent variables in the model can exert undue influence on the model (leverage).

Cook's distance summarizes information about residuals and leverage into a single statistic. Cook's distance can be calculated for each project by dropping that project and re-estimating the model without it. My statistical analysis tool does this automatically. Projects with values of Cook's distance, D, greater than $4/n$ should be examined closely (n is the number of observations). In our example, $n = 34$, so we are interested in projects for which $D > 0.118$. I find that one project, 51, has a Cook's distance of 0.147 (Example 1.28).

Example 1.28

```
. list id size effort t14 cooksd if cooksd>4/34

        id     size     effort     t14      cooksd
28.     51     1526      5931       3     .1465599
```

Why do I use Cook's distance? I use it because my statistical analysis tool calculates it automatically after ANOVA procedures. Other statistics, DFITS and Welsch distance, for instance, also summarize residual and leverage

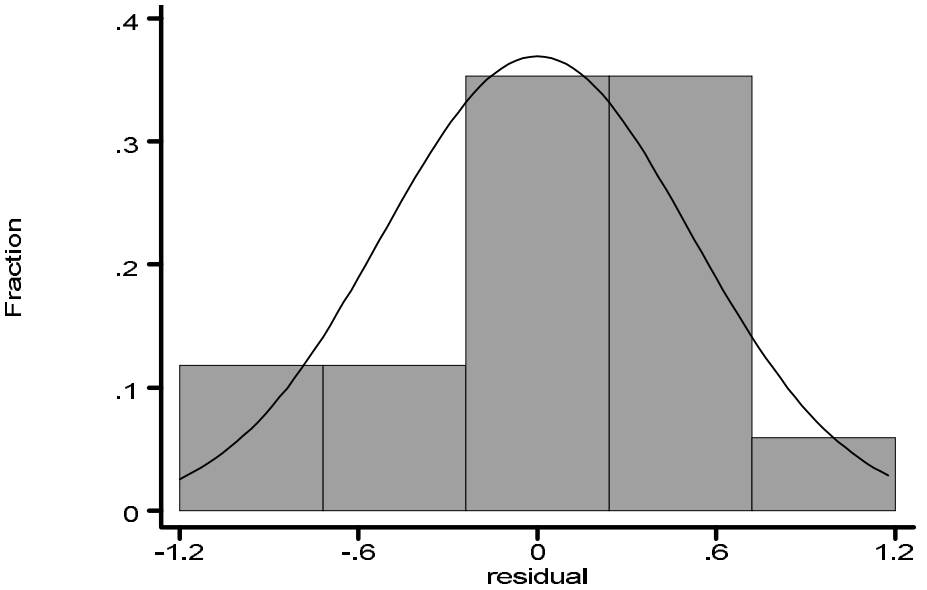

FIGURE 1.13
Distribution of residuals

information in a single value. Of course, the cut-off values are different for DIFTS and Welsh distance. Do not complicate your life; use the influence statistic that your statistical analysis tool provides.[3]

Referring back to Figure 1.8, I see that the influence of Project 51 is due to its effort being slightly low for its size compared to other projects, so it must be pulling down the regression line slightly (leverage problem). After looking closely at this project, I see no reason to drop it from the analysis. The data is valid, and given the small number of large projects we have, we cannot say that it is an atypical project. If we had more data, we could, in all likelihood, find more projects like it. In addition, 0.15 is not that far from the 0.12 cut-off value.

If a project was exerting a very high influence, I would first try to understand why. Is the project special in any way? I would look closely at the data and discuss the project with anyone who remembered it. Even if the project is not special, if the Cook's distance is more than three times larger than the cut-off value, I would drop the project and develop an alternative model using the reduced data set. Then I would compare the two models to better understand the impact of the project.

3. If you use Stata, see the "fit" procedure for these and other diagnostics.

Extracting the Equation

Our model has passed all our checks. So far, everything has been calculated automatically. We have not been forced to extract the equation and calculate effort ourselves. What is the actual equation? From the final model (Example 1.23), I see that the equation to calculate *leffort* is:

$$ln(effort) = 5.088876 + 0.7678266 \times ln(size) - 0.3856721 \times t14$$

How did I read the equation off the output? The equation is a linear equation of the form $y = a + bx_1 + cx_2$. y is $ln(effort)$, x_1 is $ln(size)$, and x_2 is $t14$. a, b, and c are the coefficients (*Coef.*) from the output. The constant (*_cons*), a, is 5.088876, the coefficient of $ln(size)$, b, is 0.7678266, and the coefficient of $t14$, c, is –0.3856721.

In a presentation or report, I give the results in the form of an equation for *effort*, not $ln(effort)$. I find it is easier for people to understand. Keep in mind that most people don't want to know how you analyzed the data or the equation; they just want to know the management implications. I almost never include an equation in an oral presentation. By all means, prepare some slides about the methodology and the equation, but do not show them unless specifically asked to go into the details in public.

To transform $ln(effort)$ into *effort*, I take the inverse natural log (or e) of each side of the equation. To do this accurately, I use all seven significant digits of the coefficients from the output. However, when I present the equation, I round the transformed coefficients to four digits. This results in about a 0.025% difference in total predicted *effort* (between a one- to two-hour difference) in this example compared with using the seven-digit coefficients. Rounding the coefficients to two digits resulted in a 100-hour difference in predicted *effort* for some projects in this sample, which I consider unacceptable. If I were to use the equation in practice to calculate *effort*, I would retain all seven significant digits. Try to always simplify as much as possible what you present to others, but be sure to use all the accuracy of the initial equations for your own calculations.

$$effort = 162.2074 \times size^{0.7678} \times e^{-0.3857 \times t14}$$

To prove to yourself that these two equations are the same, transform the *effort* equation back to the initial $ln(effort)$ equation by taking the ln of both sides and applying the following three rules from algebra:

$$ln(xyz) = ln(x) + ln(y) + ln(z), \quad ln(x)^a = aln(x), \quad \text{and} \quad ln(e) = 1$$

In Chapters 3, 4, and 5, you will see how to extract the equation from models that include categorical variables. The impact of categorical variables in an equation is simply to modify the constant term (a).

Final Comments

Now that you've learned the basics of my methodology for analyzing software project data, you are ready to attack some more complicated databases. In the following four case studies (Chapters 2-5), you will learn how to deal with common problems that occur when analyzing software project data (see Table 1.3). You will also learn how to interpret data analysis results and turn them into management implications.

TABLE 1.3
Common Problems and Where to Learn How to Deal with Them

	Chapter 2 Productivity	Chapter 3 Time to Market	Chapter 4 Development Cost	Chapter 5 Maintenance Cost
Detecting invalid data	X			
Transforming data before use	X			X
Categories with too few observations	X		X	X
Outliers	X			
Choice of best model not obvious		X	X	X
Relationships that don't make sense	X	X	X	
Confounded categorical variables				X
Choosing baseline categorical variables				X
Influential observations	X	X	X	X