18 November 2011

Chapter 6
Analyzing software measurement data

Including contributions by Barbara Kitchenham and Shari Lawrence Pfleeger from previous editions

In Chapter 4, we introduced general techniques for designing empirical studies. Then, in Chapter 5, we explained the importance of good data-gathering to support rigorous investigation. In this chapter, we turn to the analysis of the data.

Data analysis involves several activities and assumptions:

- We have a number of measurements of one or more attributes from a number of software entities (products, processes or resources). We call the set of measurements a *dataset* or a *batch*.
- We expect the software items to be *comparable* in some way. For example, we may compare modules from the same software product by examining the differences or similarities in the data. Or, we may look at several tasks from the same project to determine what the data tell us about differences in resource requirements. Similarly, we can compare several projects undertaken by the same company to see if there are some general lessons we can learn about quality or productivity.
- We wish to determine the *characteristics* of the attribute values for items of the same type (usually we look at measures of central tendency and measures of dispersion), or the *relationships* between attribute values for software items of the same or different types.

To perform the analysis, we use statistical techniques to describe the distribution of attribute values, as well as the relationship between or among attributes. That is, we want to know what a picture of the data looks like: how many very high values, how many very low values, and how the data progress from low to high. And we want to know if we can express relationships among some attribute values in a mathematical way. Thus, the purpose of the data analysis is to make any patterns or relationships more visible, so that we can use the patterns and relationships to make judgments about the attributes we are measuring.

We begin our discussion of analysis by examining the role of statistical distributions and hypothesis testing in data analysis. Then we describe several classical statistical techniques. We explain why some statistical methods must be treated with caution when we apply them to software measurement data. Then, we describe some simple methods for exploring software measurements. In the next section, we turn to the relationships among attributes and how we can use analysis techniques to describe them. Several advanced techniques can be helpful in understanding software, and we discuss three of them: multi-attribute utility theory (including the analytical hierarchy process), outranking methods, and the Bayesian approach to evaluating multiple hypotheses. We end the chapter with a reminder of how statistical tests relate to the number of groups you are analyzing.

6.1. Statistical distributions and hypothesis testing

After we have collected data from an empirical study we will use the distribution of data to evaluate the hypotheses and to make decisions and predictions. We do this in order to better understand principles of software engineering or to make decisions relevant to a practical software engineering problem. As before, you should use your goal to guide the data analysis process. The collected data generally is a sample, and represents the best available information about the status of the variables involved in the empirical study. There is always uncertainty about the relationship between the collected data and the actual relationships, thus we use probabilistic reasoning. We examine probability distributions and their role in evaluating hypotheses.

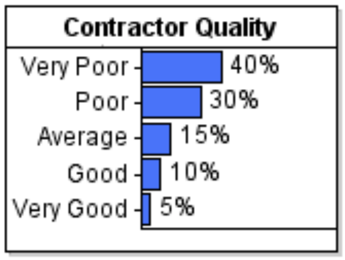
6.1.1 Probability Distributions

Consider the experiment of selecting a contractor to develop a software system. We are interested in the quality of the contractor and therefore consider the set of possible outcomes to be:

{very poor, poor, average, good, very good}

On the basis of our previous experience with contractors, or purely based on subjective judgment, we might assign the probabilities to these outcomes as shown in the table of **Figure 6.1**(a).

Very poor	0.4
Poor	0.3
Average	0.15
Good	0.1
Very good	0.05



(a) Probability table (b) Distribution shown as graph

Figure 6.1 Probability distribution

Since the numbers are all between 0 and 1 and since they sum to 1 this assignment is a valid probability measure and represents the likelihood of randomly selecting a contractor with each outcome level.

A table like the one in **Figure 6.1**(a), or an alternative graphical representation of it like the one in **Figure 6.1**(b) is called a *probability distribution*. For experiments with a discrete set of outcomes it is defined as follows:

> A *probability distribution* for an experiment is the assignment of probability values to each of the possible outcomes.

Once we have the probability distribution defined we can easily calculate the probability of any event. The *probability of an event E* is simply the sum of the probabilities of the individual outcomes that make up the event E.

> **Example 6.1** In the above experiment of selecting a contractor, suppose E is the event "quality is at least average" then E consists of the following set of outcomes:
>
> {average, good, very good}
>
> And so, from the table in **Figure 6.1**:
>
> P(*A*) = P(average) + P(good) + P(very good) = 0.15 + 0.1 + 0.05 = 0.3

The expression P(*A*) can refer to both the probability of an event *A* and the probability distribution for an experiment *A,* depending on what *A* is. So far we have only used P(*A*) to denote the probability of an event A. However, if A is itself the experiment then we will also use P(*A*) as shorthand for the full probability distribution for the experiment *A*. In the above experiment A of selecting a contractor, we use P(*A*) as shorthand for the following probability distribution:

Very poor	0.4
Poor	0.3
Average	0.15
Good	0.1
Very good	0.05

The notation is a legacy from when discussions revolved primarily around experiments that simply had two outcomes yes and no (or equivalently true and false). In such cases the experiment is simply named after the 'yes' outcome, so the following are examples of experiment names:

- "Guilty": this is the experiment whose outcome is 'yes' when a person tried is guilty and no when not guilty.
- "Disease": this is the experiment whose outcome is 'yes' when a person has the disease and no when the person does not have disease.
- "Test positive": this is the experiment whose outcome is 'yes' when a person tests positive for the disease and no when the person tests negative

So if we say P(Guilty) = 0.9 then this genuinely has the two meanings. On the one hand this means the probability of the event "guilty=yes" is 0.9. On the other hand, since it tells us that the probability of the event "guilty=yes" is 0.9, it follows that we also know the whole probability distribution, since P(yes)= 0.9 and P(no)=0.1.

In many situations assumptions about the underlying experiment enable us to automatically define the whole probability distribution rather than having to assign individual probabilities to each outcome. For example, if we are rolling a die, we can assume that each of the 6 outcomes is equally likely. In general if there are n equally likely outcomes, then the probability of each outcome is simply 1/*n*. Such a probability distribution is called the *uniform* distribution.

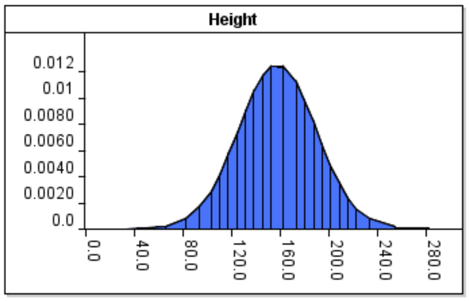
For experiments with a continuous set of outcomes, such as measuring the height of a person in centimetres, it is meaningless to assign probability values to each possible outcome. Instead, we do one of the following:

1. Divide the continuous range of outcomes into a set of discrete intervals (this process is called *discretisation*) and assign probability values to each interval. For example, if the continuous range is 0 to infinity we might define the discrete set as:

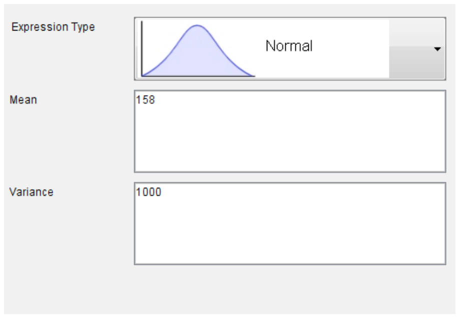
   [0, 100), [100, 110), [110, 120), [120, 130), [130, 140), [140, 150), [150, infinity)

2. Use a continuous function whose area under the curve for the range is 1. One common example of this is the Normal distribution. For example, we could use a Normal distribution such as shown in **Figure 6.2**(a) for our height

experiment (in a tool like AgenaRisk[1] you simply enter the function as shown in **Figure 6.2**(b); note that the *variance* is simply the square of the standard deviation – it is more usual to specify the variance rather than the standard deviation). Although the graph plotted here is shown only on the range 0 to 300, the Normal distribution extends from minus infinity to plus infinity. Thus, formally a Normal distribution implies that there is a 'probability' of negative values, which is really zero for height measurements. This is one reason why the Normal distribution can only be a crude approximation to the 'true' distribution.



(a) Normal distribution (with mean 158 and variance 1000) displayed as a graph


Expression Type: Normal
Mean: 158
Variance: 1000

(b) Defining the distribution in AgenaRisk

Figure 6.2 A continuous probability distribution

If you are using a continuous function as a probability distribution you need to understand that, unlike discrete probability distributions, it is not meaningful to talk about the probability of a (point) value. So, we cannot talk about the probability that a person's height is 158 cm. Instead we always need to talk in terms of a range, such as the probability that a person's height is between 157.9 and 158.1 cm. The probability is then the proportion of the curve lying between those two values. This is shown in Example 6.2, which also describes the important example of the continuous uniform distribution.

Example 6.2: Suppose that we have modelled the outcomes of measuring people's height using the Normal distribution shown in Figure 6.3.

[1] http://www.agenarisk.com/

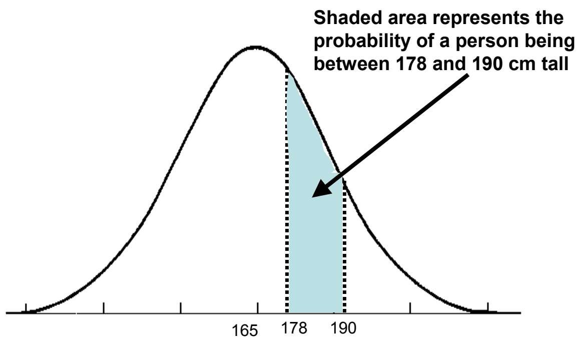


Figure 6.3 probabilities as area under the curve

The model is an idealised distribution for the underlying population. We cannot use the model to determine the probability that a person has some exact (point value), but we can use it to calculate the probability that a person's height is within any non-zero length range. We do this by computing the area under the curve between the endpoints of the range. So in the figure the shaded area is the probability that a person is between 178 and 190 cm tall. In general the area under a curve is calculated by integrating the mathematical function that describes the distribution. Even for experienced mathematicians this can be very difficult, so they use statistical tables or tools like Excel or AgenaRisk to do it for them.

6.1.2 Hypothesis testing approaches

Empirical studies are generally conducted to evaluate hypotheses. The goal is to determine which one out of a set of competing hypotheses is most plausible. By *plausible*, we mean which hypothesis has the greatest probability of being true given the data that we have gathered. As described in Chapter 4, the traditional approach is to develop a null hypothesis, along with an alternative hypothesis, or multiple alternative hypotheses. A key criterion for testing a hypothesis is a "test of significance", which evaluates the probability that a relationship was due to chance. [Fisher 1925]

The classical approaches examine whether or not the null hypothesis can be refuted with some predetermined *confidence level*, often .05. Using the .05 confidence level, we can refute the null hypothesis only if our evidence is so strong that there is only a probability of .05 (5%) that, in spite of an apparent relationship, the null hypothesis is really true.

Example 6.3 Consider the following hypotheses:

- H_0: The use of a test-first methodology during unit testing does not affect the fault densities found during system testing.
- H_1: The use of a test-first methodology during unit testing reduces the fault densities found during system testing.

> Assume that a study was conducted comparing the use of a test-first methodology to the prior method used for unit testing. The collected data shows that the mean fault densities obtained during system testing are lower for the code developed by teams employing a test-first methodology. Using the .05 confidence level, we reject H_0 only if statistical tests show that there is less than a 5% chance that H_0 is really true, due to the randomness in selecting program units or teams or some other factor relevant to the study. We do not even consider H_1 if the confidence level is greater than .05.

Using a classical approach, we do not evaluate hypotheses in terms of the magnitude of the outcome differences between the treatment and control group. Rather, we accept or reject hypotheses only on the basis of a pre-determined confidence level. Suppose the results described in Example 6.3 found that the fault densities of the treatment group (modules developed using a test-first methodology) averaged less than 50% of the fault densities of the control group (modules not developed using test-first methodology), but the statistical confidence level was .07. The results would not be significant and we would not even consider H_1, even though there was a large magnitude difference between the treatment and control groups.

The key issue is that the classical approach focuses exclusively on the precision of the results rather than the magnitude of the differences. According to Ziliak and McClosey, "statistical significance should be a tiny part of an inquiry concerned with the size and importance of a relationship." [Ziliak and McCloskey 2008] This emphasis on precision over magnitude has limited the benefits of empirical research. As a result, critical information that can aid decision makers is often discarded as insignificant. We will show in Chapter 7 that an alternative approach to data analysis is often more appropriate than the classical approach. This alternate approach employs causal models and Bayesian statistics.

The classical approach remains the predominant way that researchers analyze data. Thus, we present classical techniques in the following sections. Chapter 7 will present data analysis using causal models and Bayesian statistics.

6.2. Classical Data Analysis Techniques

After you have collected the relevant data (based on the framework we presented in earlier chapters), you must analyze it appropriately. This section describes what you must consider in choosing a classical data analysis technique. We discuss typical situations in which you may be performing a study, and what technique is most appropriate for each situation. Specific statistical techniques are described and used in the discussion. We assume that you understand basic statistics, including the following notions:

- measures of central tendency
- measures of dispersion

- distribution of data
- Student's *t*-test
- *F*-statistic
- Kruskal-Wallis test
- level of significance
- confidence limits

Other statistical tests are described here in overview, but the details of each statistical approach (including formulae and references to other statistical textbooks) can be found in standard statistical textbooks [Caulcutt 1991], [Chatfield 1998], [Dobson 2008], [Draper 1998], [Ott and Longnecker 2008]. However, you need not be a statistical expert to read this chapter; you can read about techniques to learn of the issues involved and types of problems addressed. Moreover, many of the commonly used spreadsheet and statistical packages analyze and graph the data automatically; your job is to choose the appropriate test or technique.

There are three major items to consider when choosing analysis techniques: the nature of the data you collected, why you performed the study, and the study design. We consider each of these in turn.

6.2.1 The nature of the data

In previous chapters, we have considered data in terms of its measurement scale and its position in our entity-and-attribute framework. To analyze the data, we must also look at the larger population represented by the data, as well as the distribution of that data.

6.2.1.1 Sampling, population and data distribution

The nature of your data will help you to decide what analysis techniques are available to you. Thus, it is very important for you to understand the data as a sample from a larger population of all the data you could have gathered, given infinite resources. Because you do not have infinite resources, you are using your relatively small sample to generalize to that larger population, so the characteristics of the population are important.

From the sample data, you must decide whether measured differences reflect the effects of the independent variables, or whether you could have obtained the result solely through chance. To make this decision, you use a variety of statistical techniques, based in large degree on the sample size and the data distribution. That is, you must consider the large population from which you could have selected experimental subjects and examine how your smaller sample relates to it.

In an experiment where we measure each subject once, the sample size is simply the number of subjects. In other types of experiments where we measure repeatedly (that is, the repeated-measures designs discussed in Chapter 4), the sample size is the number of times a measure is applied to a single subject. The larger the sample size, the more confident we can be that observed differences are due to more than just chance variation. As we have seen in Chapter 4, experimental error can affect our results, so we try to use large samples to minimize that error and increase the likelihood that we are concluding correctly from what we observe. In other words, if the sample size is large enough, we have confidence that the sample of measurements adequately represents the population from which it is drawn.

Thus, we must take care in differentiating what we see in the sample from what we believe about the general population. *Sample statistics* describe and summarize measures obtained from a finite group of subjects, while *population parameters* represent the values that would be obtained if all possible subjects were measured. For example, we can measure the productivity of a group of programmers before and after they have been trained to use a new programming language; an increase in productivity is a sample statistic. But we must examine our sample size, population distribution, experimental design, and other issues before we can conclude that productivity will increase for any programmers using the new language.

As we have seen in Chapter 2, we can describe the population or the sample by measures of central tendency (mean, median and mode) and measures of dispersion (variance and standard deviation). These characteristics tell us something about how the data are distributed across the population or sample. Many sets of data are distributed normally, or according to a Gaussian distribution, and have a bell-shaped curve similar to the graph shown in Figures 6.2(a) and 6.3. By definition, the mean, median and mode of such a distribution are all equal, and 96% of the data occurs within three standard deviations of the mean.

For example, the data represented by the histogram of Figure 6.4 is sometimes called "normal" because it resembles the bell-shaped curve. As the sample gets bigger, we would expect its graph to look more and more bell-shaped.

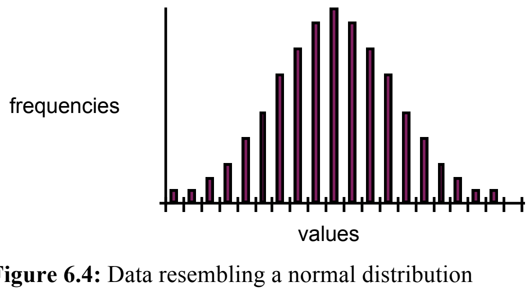


Figure 6.4: Data resembling a normal distribution

You can see in the figure that the data are evenly distributed about the mean, which is a significant characteristic of the normal distribution. But there are other distributions where the data are skewed, so that there are more data points on one side of the mean than another. For example, as you can see in Figure 6.5, most of the data are on the left-hand side, and we say that the distribution is skewed to the left.

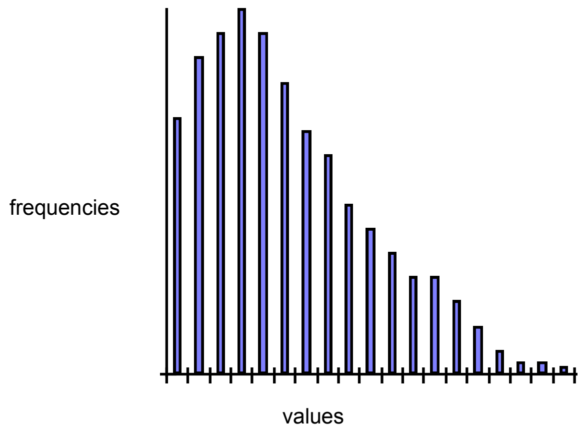


Figure 6.5: Distribution where data are skewed to the left

There are also distributions that vary radically from the bell-shaped curve, such as the one shown in Figure 6.6. As we shall see, the type of distribution determines the type of analysis we can perform.

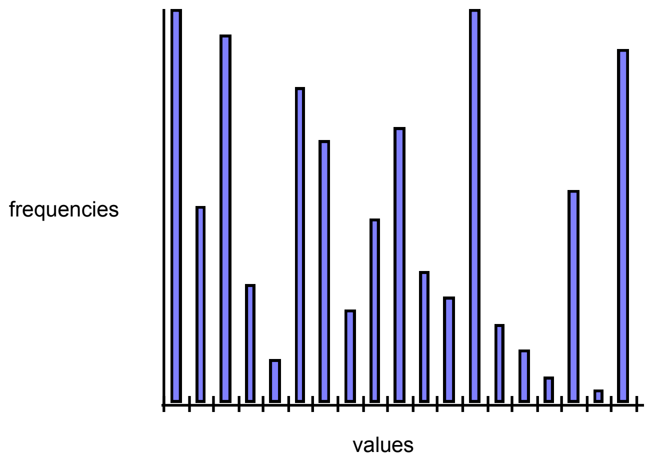


Figure 6.6: Non-normal distribution

6.2.1.2 The distribution of software measurements

Let us see how common software engineering data "measures up" to being normally distributed. In earlier chapters, we mentioned that many common statistical operations and tests are not meaningful for measures that are not on an interval or ratio scale. Unfortunately, many software measures are only ordinal. This scale results from our wanting to use measures that categorize and rank. For example, we create categories of failure severity, so that we can tell if most of our system failures are trivial, rather than life threatening. Similarly, we ask our users to rate their satisfaction with the system, or ask designers to assign a quality measure to each requirement before design begins. Such ordinal measures do not allow us to calculate means and standard deviations, as those are suitable only for interval, ratio, and absolute data. Thus, we must take great care in choosing analysis techniques that are appropriate to the data we are collecting. (And as we saw in Chapter 5, we may collect data based on the type of analysis that we want to do.)

In addition to measurement scale considerations, we must also consider how the data are gathered. As we have seen in Chapter 4, many statistical tests and analysis techniques are based on the assumption that datasets are made up of measurements drawn at random. Indeed, we design our data collection techniques to encourage this randomization, although it is not always possible. Moreover, even when software measurements are on a ratio scale and are selected randomly, the underlying distribution of the general population of data is not always normal. Frequently, the datasets are discrete and non-negative, skewed (usually towards the left), and usually include a number of very large values.

Let us examine these problems more carefully by looking at examples of two real datasets.

Table 6.1a: Dataset 1

Project effort (months)	*Project duration (months)*	*Product size (lines of code)*
16.7	23.0	6050
22.6	15.5	8363
32.2	14.0	13334
3.9	9.2	5942
17.3	13.5	3315
67.7	24.5	38988
10.1	15.2	38614
19.3	14.7	12762
10.6	7.7	13510
59.5	15.0	26500

Table 6.1b: Dataset 2

Module size	*Module fan-out*	*Module fan-in*	*Module control flow paths*	*Module faults*
29	4	1	4	0
29	4	1	4	2
32	2	2	2	1

33	3	27	4	1
37	7	18	16	1
41	7	1	14	4
55	1	1	12	2
64	6	1	14	0
69	3	1	8	1
101	4	4	12	5
120	3	10	22	6
164	14	10	221	11
205	5	1	59	11
232	4	17	46	11
236	9	1	38	12
270	9	1	80	17
549	11	2	124	16

Example 6.4: Consider the datasets in Tables 6.1a and 6.1b. Both contain measures of project and product information from commercial software systems. Dataset 1 includes all of the data that was available from a particular environment during a particular time interval; it is typical of the project-level data used for project cost control. Dataset 2 reflects attributes of all the procedures in a particular product subsystem and is typical of component-level data. This second dataset includes various internal product measures that will be described in detail in Chapter 9.

In a normal distribution, the mean, median and mode of the data are the same. We can use this information to tell us whether the example datasets are normally distributed or not. If, for each attribute of Table 6.1, we compare the mean with the median, we see in Table 6.2 that the median is usually considerably smaller than the mean. Thus, the data are not normally distributed.

Table 6.2a: Summary statistics for dataset 1

Statistic	*Effort*	*Duration*	*Size*
Mean	26.0	15.2	16742
Median	18.3	14.8	13048
Standard deviation	21.3	5.1	13281

Table 6.2b: Summary statistics for dataset 2

Statistic	*Size*	*Fan-out*	*Fan-in*	*Paths*	*Faults*
Mean	133.3	5.6	5.8	40	5.9
Median	69	4	1	14	4.0
Standard deviation	135.6	3.5	7.9	57.0	5.8

Many statistical references describe other techniques for assessing whether a distribution is normal. Until we know something about our data, we must be very cautious about the use of techniques that assume an underlying normal distribution. When we do not know anything about the distribution, there are a number of approaches to dealing with our lack of knowledge:

- We can use robust statistics and non-parametric methods. *Robust statistical methods* are descriptive statistics that are resilient to non-normality. That is, regardless of whether the data are normally distributed or not, robust methods yield meaningful results. On the other hand, *non-parametric statistical techniques* allow us to test various hypotheses about the dataset without relying on the properties of a normal distribution. In particular, non-parametric techniques often use properties of the *ranking* of the data.

- We can attempt to transform our basic measurements into a scale in which the measurements conform more closely to the normal distribution. For example, when investigating relationships between project effort and product size, it is quite common to transform to the logarithmic scale. Whereas the original data are not normally distributed, the logarithms of the data are.
- We can attempt to determine the true underlying distribution of the measurements and use statistical techniques appropriate to that distribution.

Example 6.5: Mayer and Sykes looked at the relationship between lines of code per module and the number of decisions contained in each module. They found a very good fit for two different datasets using the negative binomial distribution. [Mayer and Sykes 1989]

6.2.1.3 Statistical inference and classical hypothesis-testing

Classical hypothesis testing, introduced in Section 6.1.2, makes use of statistical inference. The distribution type plays a big part in how we make inferences from our data. Statistical inference is the process of drawing conclusions about the population from observations about a sample. The process and its techniques depend on the distribution of the data. As we have noted above, parametric statistical techniques apply only when the sample has been selected from a normally distributed population; otherwise, we must use non-parametric techniques. In both cases, the techniques are used to determine if the sample is a good representation of the larger population. For example, if we perform an experiment to determine the increase in productivity of programmers, as described in Example 4.15, we can calculate a mean productivity and standard deviation. Statistical inference tells us whether the average productivity for any programmer using the new language is likely to be the same as the average productivity of our sample.

The logic of statistical inference is based on the two possible outcomes that can result from any statistical comparison:

1. The measured differences observed in the course of the experiment reflect simple chance variation in measurement procedures alone (that is, there is no real difference between treated subjects and untreated ones), or
2. The measured differences indicate the real treatment effects of the independent variable(s).

The first case corresponds to our statement of the null hypothesis; there is no change. Statisticians often denote the null hypothesis as H_0. The second case is the alternative hypothesis, often written as H_1. The purpose of statistical analysis is to see whether the data justify rejecting H_0. The rejection of H_0 does not always mean the acceptance of H_1; there may be several alternative hypotheses, and rejection of H_0 means simply that more experimentation is needed to determine which alternative hypothesis is the best explanation of the observed behavior.

We emphasize that statistical analysis is directed only at whether we can reject the null hypothesis. In this sense, our data can refute the alternative hypothesis in light of empirical evidence (that is, the data support the null hypothesis because there is no compelling evidence to reject it), but we can never prove it. In many sciences, a large body of empirical data is amassed, wherein each case rejects the same null hypothesis; then, we say loosely that this evidence "confirms," "suggests" or "supports" the alternative hypothesis, but we have not proven the hypothesis to be fact.

The statistical technique applied to the data yields the probability that the sample represents the general population; it provides the confidence we can have in this result and is called the *statistical significance* of the test. Usually referred to as the alpha (α) level, acceptable significance is agreed upon in advance of the test and often α is set at 0.05 or 0.01. That is, an experiment's results are not considered to be significant unless we are sure that there is at least a 0.95 or 0.99 probability that our conclusions are correct.

Of course, there is no guarantee of certainty. Accepting the null hypothesis when it is actually false is called a *Type II error*. Conversely, rejecting the null hypothesis when it is true is a *Type I error*. Viewed in this way, the level is the probability of committing a Type I error, as shown in Table 6.3.

Table 6.3: Results of hypothesis testing

State of the world	*Decision: Accept H_0*	*Decision: Reject H_0*
H_0 is true	Correct decision Probability = 1 - a	Type I error Probability = a
H_0 is false	Type II error Probability = a	Correct decision Probability = 1 - a

We have seen how a normal distribution is continuous and symmetric about its mean, yet software data are often discrete and not symmetric. If you are not sure whether your data are normal or not, you must assume that they are not, and use techniques for evaluating non-normal data.

In the examples that follow, we consider both normal and non-normal cases, depending on the type of data. The examples mention specific statistical tests, descriptions of which are at the end of this chapter, and additional information can be found in standard statistical textbooks. Many of these statistical tests can be computed automatically by spreadsheets, so you need not master the underlying theory to use them; it is important only to know when the tests are appropriate for your data.

6.2.2 Purpose of the experiment

In Chapter 4, we noted two major reasons to conduct a formal investigation, whether it is an experiment, case study or survey:

- To confirm a theory
- To explore a relationship

Each of these requires analysis carefully designed to meet the stated objective. In particular, the objective is expressed formally in terms of the hypothesis, and the analysis must address the hypothesis directly. We consider each one in turn, mentioning appropriate analysis techniques for each objective; all of the techniques mentioned in this section will be explained in more detail later in this chapter.

6.2.2.1 Confirming a theory

Your investigation may be designed to explore the truth of a theory. The theory usually states that use of a certain method, tool, or technique (the treatment) has a particular effect on the subjects. The theoretical effect is to improve the process or product in some way compared to another treatment (usually the existing method, tool or technique). For example, you may want to investigate the effect of a test-first methodology by comparing it with your existing testing methods. The usual analysis approach for this situation is *analysis of variance*. That is, you consider two populations, the one that uses the old technique and the one that uses the new, and you do a statistical test to see if the difference in treatment results is statistically significant. You analyze the variance between the two sets of data to see if they come from one population (and therefore represent the same phenomenon) or two (and therefore may represent different phenomena). The first case corresponds to accepting the null hypothesis, while the second corresponds to rejecting the null hypothesis. Thus, the theory is not proven; instead, the second case provides empirical evidence that suggests some reason for the difference in behavior.

There are two cases to consider: normal data and non-normal data. If the data come from a normal distribution and you are comparing two groups, you can use tests such as the *Student's t-test* to analyze the effects of the two treatments. If you have more than two groups to compare, a more general analysis of variance test, using the *F statistic*, is appropriate. Both of these are described in most statistics books.

Example 6.6: You are investigating the effect on productivity of the use of a new tool. You have two groups that are otherwise equal except for use of the tool: group *A* is using the existing method (without the tool), while group *B* is using the tool to perform the designated task. You are measuring productivity in terms of thousands of delivered source code instructions per month, and the productivity data come from a normal distribution.

You can use a Student's *t*-test to compare group *A*'s productivity data with group *B*'s to see if the use of the tool has made a significant change in productivity.

Example 6.7: On the other hand, suppose you want to investigate whether the test-first technique yields higher-quality code than your current testing technique. Your null hypothesis is stated as

> Code developed and tested using the test-first technique has the same number of defects per hundred lines of code as code developed using current testing techniques.

You collect data on number of defects per line of code (a measures of defect density) for each of two groups, and you seek an analysis technique that will tell you whether or not the data support the hypothesis. Here, the data on defects per line of code are not normally distributed. You can analyze the defect data by ranking it (for example, by ranking modules according to their defect density) and using the *Kruskal-Wallis test* to tell you if the mean rank of the test-first modules is lower than that of the non-test-first data.

6.2.2.2 Exploring a relationship

Often, an investigation is designed to determine the relationship among data points describing one variable or across multiple variables. For example, you may be interested in knowing the normal ranges of productivity or quality on your projects, so that you have a baseline to compare for the future. A case study may be more appropriate for this objective, but you may want to answer this question as part of a larger experiment. Several techniques can help to to answer questions about a relationship: box plots, bar charts, control charts, scatter plots (or scatter diagrams), and correlation analysis.

A *box plot* can depict a summary of the range and distribution of a set of data for one variable. It shows where most of the data are clustered and the location of outlier data. A *bar* chart provides an alternative way to display a single variable. Bar charts are especially useful when you are comparing the data from a small number of identified entities. A *control chart* shows the trends of a variable over time, and can help you to spot occurrences of abnormal data values. While box plots, bar charts, and control charts show information about one variable, a *scatter plot* depicts the relationship between two variables. By viewing the relative positions of pairs of data points, you can visually determine the likelihood of an underlying relationship between the variables. You can also identify data points that are atypical, because they are not organized or clustered in the same way as the other data points.

Correlation analysis goes a step further than a scatter diagram by using statistical methods to confirm whether there is a relationship between two attributes. Correlation analysis can be done in two ways: by generating measures of association that indicate the closeness of the behavior of the two variables, or by generating an equation that describes that behavior.

6.2.3 Decision tree

To help you understand when to use one analysis technique or another, we have constructed a decision tree, shown in Figure 6.7, to take into account the major considerations discussed in this section. The decision tree is to be read from left to right. Beginning with the objective of your investigation, move along the branch that fits your situation until you reach a leaf node with the appropriate analysis technique(s). The next section provides further details concerning these analysis techniques.

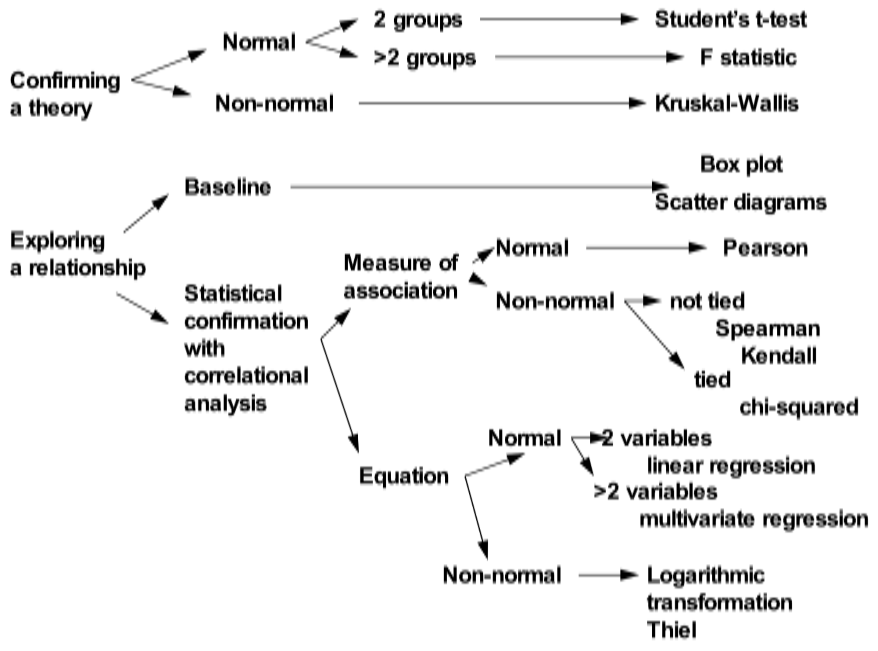


Figure 6.7: Decision tree for analysis techniques

6.3. Examples of simple analysis techniques

There are many robust techniques that are useful with software measurement data, regardless of the distribution. You need not be a statistician to understand and use them, and you can implement them using simple spreadsheets or statistical packages. The previous section outlined several approaches, based on the goals of your investigation. In this section, we look at some of the techniques in more detail.

6.3.1 Box plots

Software measurement datasets are often not normally distributed, and the measurements may not be on a ratio scale. Thus, you should use the median and quartiles to define the central location and spread of the component values, rather than the more usual mean and variance. These robust statistics can be presented in a visual form called a *box plot*, as shown in Figure 6.8. Box plots are constructed from three summary statistics: the median, the upper quartile and the lower quartile.

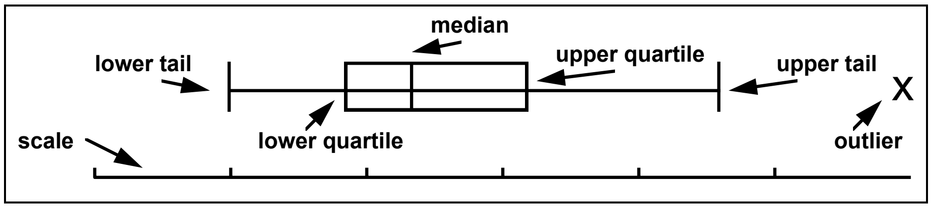


Figure 6.8: Drawing a box plot

The *median* is the middle-ranked item in the dataset. That is, the median is the value *m* for which half the values in the dataset are larger than *m* and half are smaller than *m*. The *upper quartile u* is the median of the values that are more than *m*, and the *lower quartile l* is the median of the values that are less than *m*. Thus, *l, m* and *u* split the dataset into four parts. We define the *box length*, *d*, to be the distance from the upper quartile to the lower; thus, $d = u — l$. Next, we define the *tails* of the distribution. These points represent the theoretical bounds between which we are likely to find all the data points if the distribution is normal. If the data is on an interval, ratio or absolute scale, the theoretical upper tail value is the point $u + 1.5d$, and the lower tail value is $l — 1.5d$. These theoretical values must then be truncated to the nearest actual data point to avoid meaningless concepts (such as negative lines of code) and to demonstrate the essential asymmetry of skewed datasets. Values outside the upper and lower tails are called *outliers*; they are shown explicitly on the box plot, and they represent data points that are unusual in some way.

The relative positions of the median, the quartiles, and the tails in a box plot can indicate if the dataset is skewed. If the dataset is symmetric about the median, the median will be positioned in the center of the box, and the tail lengths (that is, the length from the quartile to the tail point) will be equal. However, if the dataset is skewed, the median will be offset from the center of the box and the tail lengths will be unequal.

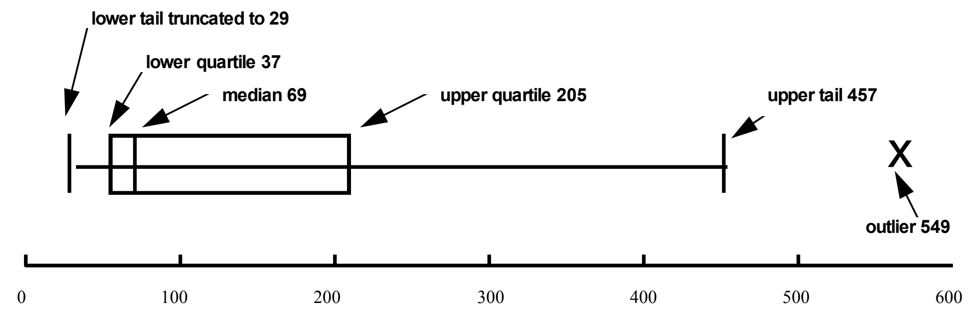


Figure 6.9: Box plot of lines of code (17 procedures) for dataset 2 of Table 6.1b

To see how box plots are used, we apply them to dataset 2 of Table 6.1. Figure 6.9 shows the result.

> **Example 6.8:** The lines of code values in Table 6.1b are arranged in ascending order. The median is the ninth value: 69. The lower quartile is the fifth value: 37. The upper quartile is the thirteenth value: 205. Thus, the box length is 168. Constructing a box plot is more difficult when there is an even number of observations; in that case, you must take the average of two values when there is no "middle" value. Hoaglin describes this procedure in more detail. [Hoaglin *et al.* 2000] It is clear that the box plot in Figure 6.9 is strongly skewed to the left.

Box plots are simple to compute and draw, and they provide a useful picture of how the data are distributed. The outliers are especially important when you are looking for abnormal behaviors.

System	KLOC	MOD	FD
A	10	15	36
B	23	43	22
C	26	61	15
D	31	10	33
E	31	43	15
F	40	57	13
G	47	58	22
H	52	65	16
I	54	50	15
J	67	60	18
K	70	50	10
L	75	96	34
M	83	51	16
N	83	61	18
P	100	32	12
Q	110	78	20
R	200	48	21

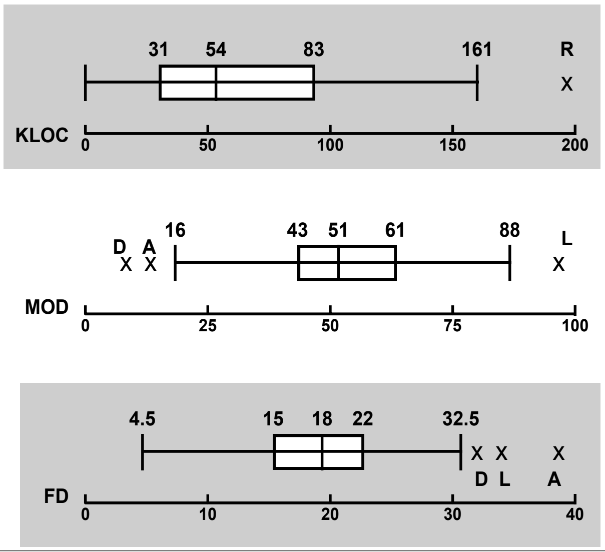


Figure 6.10: Box plots for different attributes

Example 6.9: Figure 6.10 shows box plots for measures taken on 17 software systems. Each system provided three measures: thousands of lines of code (KLOC), average module size in LOC (MOD), and the number of faults found per KLOC (fault density, or FD). The top box plot illustrates KLOC, the middle MOD, and the bottom FD. Each identifies outliers with respect to the measure it is depicting.

Notice that MOD and FD have exactly the same outliers! Systems D, L and A have abnormally high fault densities, and they also have unusual module sizes: D and A have abnormally low MOD, while L has abnormally high MOD. Further investigation must identify more clearly the relationship between MOD and FD, but the initial data seem to confirm the widely held belief that a system should be composed of modules that are neither too small nor too large.

The outliers in the KLOC box plot seem to be unrelated to those in fault density.

In general, since the outliers represent unusual behavior, quality assurance staff can use box plots to identify the modules that should be tested or inspected first. In this example, we might want to examine all systems whose MOD values are outliers in the box plot, since these modules are the ones most likely to be fault prone.

Thus, box plots point us to abnormal or unusual behavior. On the other hand, they can also help us to see what is usual or normal.

Example 6.10: We often analyze system test fault density (defined as the number of faults discovered during system test divided by some measure of product size) on past projects to

> identify the fault density norm for an organization or product line. We can calculate the likely number of system test faults expected for a new product entering system test by multiplying the median value of the fault density by the actual product size measure. Then, we can define an acceptable range to be between the upper and lower quartiles, and use it to monitor the results of system test.

Many developers seek to implement statistical quality control on software development. For this kind of control, it is too stringent to restrict the identification of potential problem components to those that are outliers. A compromise solution is to review quickly those components with values greater than the upper quartile; then, give greater scrutiny to the components with values greater than the upper quartile plus the box length. Components identified by these criteria are considered to be *anomalies* rather than outliers.

These examples show the utility of box plots in providing norms for planning purposes, summarizing actual measurements for the purposes of project monitoring, and identifying software items with unusual values for quality control and process evaluation.

6.3.2 Bar charts

We saw the utility of box plots in visualizing what is happening in a dataset. The box plot hides most of the expected behavior and shows us instead what is unusual. Another useful depiction of data is a *bar chart*. Here, we simply display all the data ordered to see what the patterns or trends may be. For example, the graph in Figure 6.11 contains a bar for each of the ten projects in dataset 1 of Table 6.1. The *x*-axis is labeled from 1 to 10 for the project number, the bar height shows the effort, and there is one bar for each measurement in the dataset.

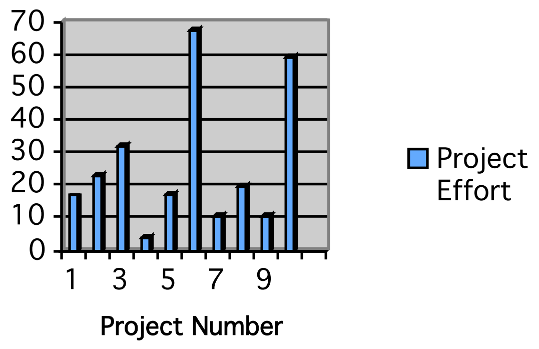


Figure 6.11: Bar chart of effort from dataset 1

We can see from the bar chart that most of the projects require less than 40 person-months of effort, but two require a great deal more. Such information raises many questions, and often we want to know the relationship between one attribute (such as effort in this example) and others (such as size). Unlike box plots, bar charts allow us to readily identify the entity associated with each measured value.

6.3.3 Control charts

Another helpful technique is a *control chart*, which helps you to see when your data are within acceptable bounds. By watching the data trends over time, you can decide whether to take action to prevent problems before they occur. To see how control charts work, consider first some non-software examples. Many processes have a normal variation in a given attribute. For instance, steel manufacturers rarely make a one-inch nail that is exactly one inch long; instead, they set tolerances, and a one-inch nail can be a very small fraction above or below an inch in length and still be acceptable. We would expect the actual length values to be randomly distributed about the mean of one inch, and 95% of the nails would have lengths falling within two standard deviations of the one-inch mean. Similarly, the voltage in a 220-volt electrical outlet is rarely exactly 220 volts; instead, it ranges in a band around 220 volts, and appliances are designed to work properly within that small band (but may fail to work if the voltage drops too low or jumps too high).

In the same way, there are parts of the software process that can be expected to behave in a random way, and we can use the control limits to warn us when a value is unusual. We use the values of two standard deviations above and below the mean as guidelines. We want to determine the reasons why any value falls outside these boundaries.

Consider the data in Table 6.4. Here, we have information about the ratio between preparation hours and inspection hours for a series of design inspections. We calculate the mean and standard deviation of the data, and then two control limits. The *upper control limit* is equal to two standard deviations above the mean, while the *lower control limit* is two standard deviations below the mean (or zero, if a negative control limit is meaningless). The control limits act as guidelines for understanding when the data are within random statistical variation and when they represent unusual behavior. In this sense, control charts are similar to box plots.

Table 6.4: Selected design inspection data [Schulmeyer and McManus 1987]

Component number	*Preparation hours/inspection hours*
1	1.5
2	2.4
3	2.2
4	1.0

5	1.5
6	1.3
7	1.0
Mean	1.6
Standard deviation	.5
Upper control limit (UCL)	2.6
Lower control limit (LCL)	.4

To visualize the behavior of the data, we construct a graph called a *control chart*. The graph shows the upper control limit, the mean, and the lower control limit. As you can see in Figure 6.12, the data are plotted so that we can see when they are getting close to or exceeding the control limits. In the figure, the data stay within the control limits. Thus, the preparation hours per hour of inspection are within what we would expect of random variation. However, if the data were to exceed the control limits, we would take action to bring them back under control — that is, back within the band between the upper and lower control limits. In this example, exceeding the upper control limit means that too much preparation is being done, or not enough time is being spent during inspection. If the value is below the lower control limit, then not enough preparation is being done, or too much time is being spent during inspection. Even when data do not exceed the control limits, action can be taken when the trend veers toward the edge of the acceptable band, so that behavior is drawn back toward the middle.

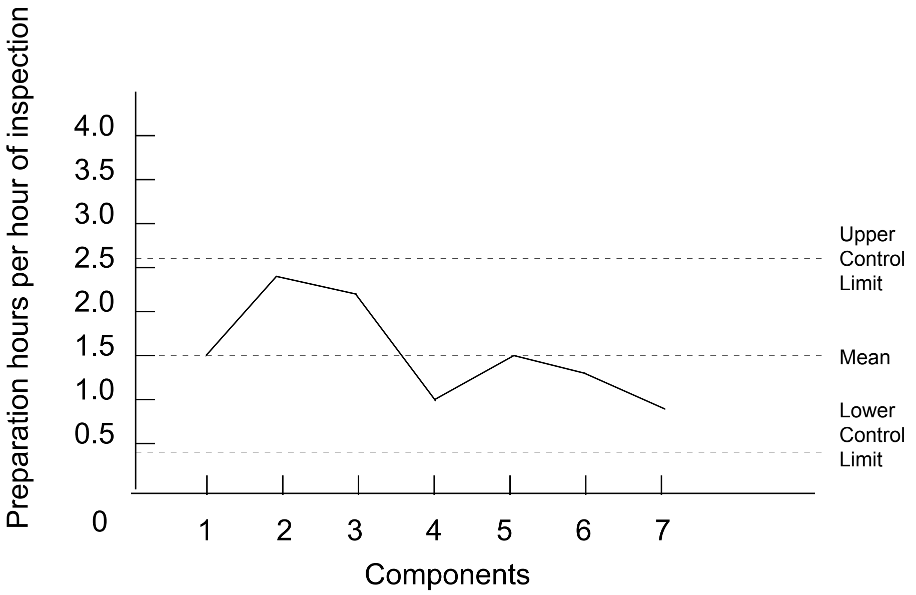


Figure 6.12: Inspection control chart showing hours of preparation per hour of inspection [Humphrey 1989]

6.3.4 Scatter plots

In Chapter 4, we saw many reasons for wanting to investigate the relationships among attribute values. For instance, understanding relationships is necessary when, for planning purposes, we wish to predict the future value of an attribute from some known value(s) of the same or different attributes. Likewise, quality control requires us to identify components having an unusual combination of attribute values.

When we are interested in the relationship between two attributes, a scatter plot offers a visual assessment of the relationship by representing each *pair* of attribute values as a point in a Cartesian plane.

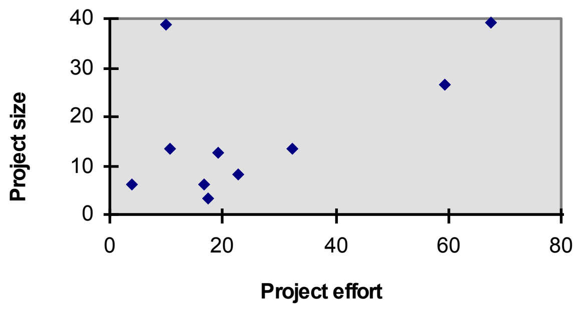


Figure 6.13: Scatter plot of project effort against project size for dataset 1

Example 6.11: Figure 6.13 is a scatter plot of effort (measured in person-months) against size (measured in thousands of lines of code) for dataset 1 in Table 6.1. Each point represents one project. The x-coordinate indicates project effort and the y-coordinate is the project size. This plot shows us that there appears to be a general relationship between size and effort, namely that effort increases with the size of the project.

However, there are a few points that do not support this general rule. For instance, two projects required almost 40 person-months to complete, but one was 10 KLOC, while the other was 68. The scatter plot cannot explain this situation, but it can suggest hypotheses that we can then test with further investigation.

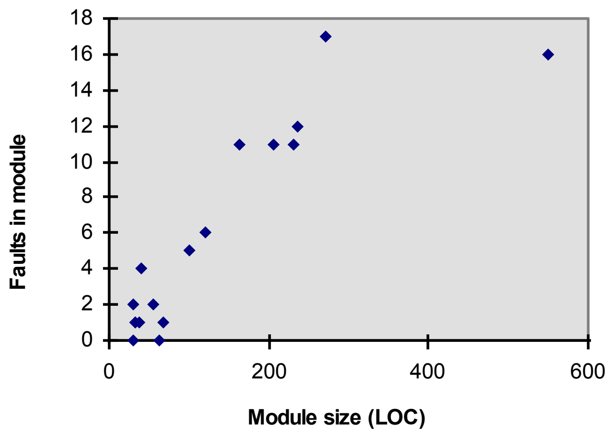


Figure 6.14: Scatter plot of module faults against module size for dataset 2

> **Example 6.12:** Similarly, Figure 6.14 is a scatter plot of the module data shown in dataset 2 of Table 6.1. Here, module size (measured in lines of code) is graphed against the number of faults discovered in the module. That is, there is a point in the scatter plot for each module in the system. Again, the plot shows a trend, in that the number of faults usually increases with the module size. However, again we find one point that does not follow the rule: the largest module does not appear to have as many faults as might have been expected.

Thus, scatter plots show us both general trends and atypical situations. If a general relationship seems to be true most of the time, we can investigate the situations that are different and determine what makes the projects or products anomalous. The general relationship can be useful in making predictions about future behavior, and we may want to generate an estimating equation to project likely events. Similarly, we may project likely anomalies to control problems by taking action to avert their behavior or by minimizing their effects.

For obvious reasons, scatter plots are not helpful for more than three variables, unless you evaluate relationships for all possible pairs and triples. Such partitioning of the problem does not give you a good overview of the behavior that interests you.

6.3.5 Measures of association

Scatter plots depict the behavior of two attributes, and sometimes we can determine that the two attributes are related. But the appearance of a relationship is not enough evidence to draw conclusions. Statistical techniques that can help us evaluate the likelihood that the relationship seen in the past will be seen again in the future. We call these techniques *measures of association*, and the measures are supported by statistical tests that check whether the association is significant.

For normally distributed attribute values, the *Pearson correlation coefficient* is a valuable measure of association. Suppose we want to examine the association between two attributes, say x and y. For instance, we saw in Example 6.12 that x could be the size of a module, while y is the number of faults found in the module. If the datasets of x and y values are normally distributed (or nearly), then we can form pairs (x_i, y_i), where there are i software items and we want to measure the association between x and y. The total number of pairs is n, and for each attribute, we calculate the mean and variance. We represent the mean of the x values by m_x, and the mean of the y values by m_y. Likewise, var(x) is the variance of the set of x values, and var(y) the variance of the y values. Finally, we calculate

$$r = \sum_{i=1}^{n} \frac{(x_i - m_x)(y_i - m_y)}{\sqrt{n \operatorname{var}(x) \operatorname{var}(y)}}$$

The value of r, called the correlation coefficient, varies from -1 to 1. When r is 1, then x and y have a perfect positive linear relationship; that is, when x increases, then so does y in equal linear steps. Similarly, -1 indicates a perfect negative linear relationship (that is, when x increases, y decreases linearly), and 0 indicates no relationship between x and y (that is, when x increases, y is just as likely to increase as to decrease). Statistical tests can be used to check whether a calculated value of r is significantly different from zero at a specified level of significance; in other words, computation of r must be accompanied by a test to indicate how much confidence we should have in the association.

However, as we have noted before, most software measurements are not normally distributed and usually contain atypical values. In fact, the dataset in Example 6.12 is not from a normal distribution, so the Pearson correlation coefficient is not recommended for it. In these cases, it is preferable to use robust measures of association and non-parametric tests of significance. One approach to the problem is the use of a robust correlation coefficient; other approaches, including contingency tables and the chi-squared test, are discussed in standard statistical textbooks.

6.3.6 Robust correlation

The most commonly used robust correlation coefficient is *Spearman's rank correlation coefficient*. It is calculated in the same way as the Pearson correlation coefficient, but the x and y values are based on ranks of the attributes, rather than raw values. That is, we place the attribute values in ascending order and assign 1 to the smallest value, 2 to the next smallest, and so on. If two or more raw values are equal, each is given the average of the related rank values.

Example 6.13: The two smallest modules in dataset 2 have 29 lines of code. In a ranking, each module is assigned the rank of 1.5, calculated as the average of ranks 1 and 2.

Kendall's robust correlation coefficient t varies from -1 to 1, as Spearman's, but the underlying theory is different. The Kendall coefficient assumes that, for each two pairs of attribute values, (x_i,y_i) and (x_j,y_j), if there is a positive relationship between the attributes, then when x_i is greater than x_j, then it is likely that y_i is greater than y_j. Similarly, if there is a negative relationship, then when x_i is greater than x_j, it is likely that y_i is less than y_j. Kendall's *t* is based on assessing all of pairs of vectors (in a dataset of points) and comparing positive with negative indications.

Kendall's correlation coefficient is important because it can be generalized to provide partial correlation coefficients. These partial values are useful when the relationship between two attributes may in fact be due to the relationship between both attributes and a third attribute. (See Seigel and Castellan's classic statistics book for an explanation of partial coefficients and their use. [Siegel and Castellan 1988])

The rank correlation coefficients are intended to be resilient both to atypical values and to non-linearity of the underlying relationship, as well as not being susceptible to the influence of very large values.

Table 6.5: Correlation coefficients for dataset 1

Type	*Effort v. Size*	*Effort v. Size (atypical items removed)*	*Effort v. Duration*	*Effort v. Duration (atypical items removed)*
Pearson	0.57	0.91	0.57	0.59
Spearman	0.46	0.56	0.48	0.50
Kendall	0.33	0.67	0.38	0.65

Table 6.6: Correlation coefficients for dataset 2

Type	*LOC v. Faults*	*LOC v. Faults (atypical removed)*	*Paths v. Faults*	*Paths v. Faults (atypical removed)*	*FO v. Faults*	*FI v. Faults*
Pearson	0.88	0.96	0.68	0.90	0.65	-0.11
Spearman	0.86	0.83	0.80	0.79	0.54	0.02
Kendall	0.71	0.63	0.60	0.62	0.58	0.02

Example 6.14: Tables 6.5 and 6.6 contain the values of the Pearson, Spearman and Kendall correlation coefficients for various attribute pairs from datasets 1 and 2 in Table 6.1. Notice that, in both tables, the robust correlations (that is, the Spearman and Kendall values) are usually less than the Pearson correlations. This difference in correlation value is caused by the very large values in both datasets; these large values tend to inflate the Pearson correlation coefficient but not the rank correlations.

Notice, too, that we have calculated some of the correlations both with and without some of the atypical values, as identified by scatter plots earlier in this chapter. It appears that the rank correlation coefficients are reasonably resilient to atypical values. Moreover, the relationship between control flow paths and faults in a module is non-linear, but the rank correlations are not affected by it.

Table 6.5, reflecting dataset 1, shows that, in general, all the correlation coefficients were susceptible to the gross outliers in the data. So although the rank correlation coefficients are more reliable for software measurement data than the Pearson correlation, they can be misleading without visual inspection of any supposed relationship.

Example 6.14 includes a practice common to data analysis: eliminating atypical data points. Sometimes, a scatter plot or correlational analysis reveals that some data illustrate behavior different from the rest. It is important to try to understand what makes these data different. For example, each point may represent a project, and the unusual behavior may appear only in those projects done for a particular customer, X. With those data points removed, the behavior of the rest can be analyzed and used to predict likely behavior on future projects done for customers other than X. However, it is not acceptable to remove data points only to make the associations look stronger; there must be valid reasons for separating data into categories and analyzing each category separately.

6.3.7 Linear regression

Having identified a relationship using box plots, scatter plots, or other techniques, and having evaluated the strength of the association between two variables, our next step is expressing the nature of the association in some way. Linear regression is a popular and useful method for expressing an association as a linear formula. The linear regression technique is based on a scatter plot. Each pair of attributes is expressed as a data point, (x_i,y_i), and then the technique calculates the line of best fit among the points. Thus, our goal is to express attribute y, the dependent variable, in terms of attribute x, the independent variable, in an equation of the form

$$y = a + bx$$

To see how this technique works, consider again the scatter plot of Figure 6.10. We can draw a line roughly to estimate the trend we see, showing that effort increases as size increases; this line is superimposed on the scatter plot in Figure 6.15.

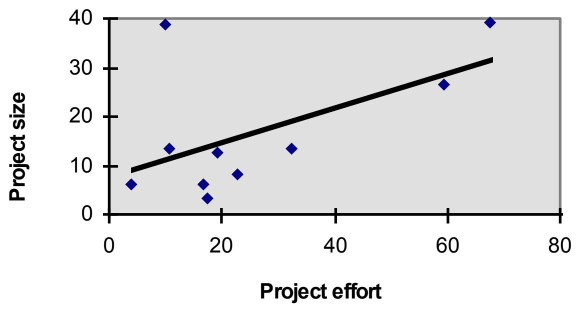


Figure 6.15: Plot of effort against size for dataset 1, including line to show trend

The theory behind linear regression is to draw a line from each data point vertically up or down to the trend line, representing the vertical distance from the data point to the trend. In some sense, the length of these lines represent the discrepancy between the data and the line, and we want to keep this discrepancy as small as possible. Thus, the line of "best fit" is the line that minimizes these distances.

The mathematics required to calculate the slope, *b*, and intercept, *a*, of this "best fit" line are straightforward. The discrepancy for each point is called the *residual*, and the formula for generating the linear regression line minimizes the sum of the squares of the residuals. We can express the residual for a given data point as

$$r_i = y_i - a - bx_i$$

Minimizing the sum of squares of the residuals leads to the following equations for *a* and *b*:

$$b = \frac{\sum (x_i - m_x)(y_i - m_y)}{\sum (x_i - m_x)^2}$$

$$a = m_y - bm_x$$

The least squares technique makes no assumptions about the normality of the distribution of either attribute. However, to perform any statistical tests relating to the regression parameters (for example, we may want to determine if the value of *b* is significantly different from 0), it is necessary to assume that the residuals are distributed normally. In addition, the least squares approach must be used with care when there are many large or atypical values; these data points can distort the estimates of *a* and *b*.

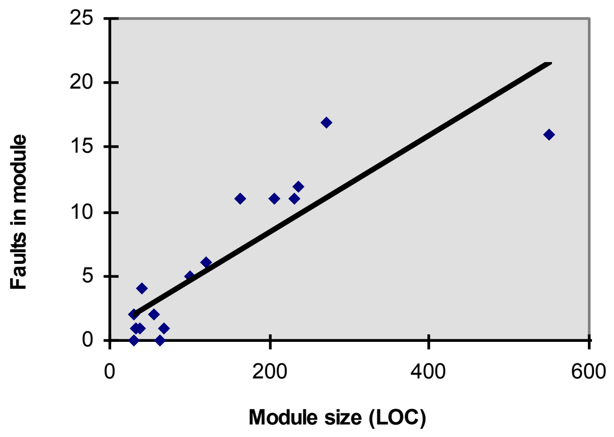


Figure 6.16: Module size against faults, including linear regression line

Example 6.15: Figure 6.16 shows the scatter plot of module size against faults from dataset 2 in Table 6.1. Imposed on the scatter plot is the line generated by linear regression. Notice that there are several values in the dataset that are far from the line, including the point representing the largest module. If we remove the data point for the largest module, the regression line changes dramatically, as shown in Figure 6.17. The remaining data points are much closer to the new line.

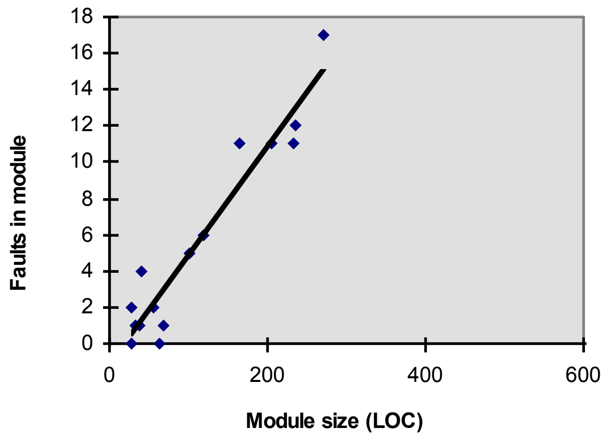


Figure 6.17: Module size against faults, using dataset 2 minus largest module

After fitting a regression line, it is important to check the residuals to see if any are unusually large. We can plot the residual values against the corresponding dependent variable in a scatter plot. The resulting graph should resemble an ellipsoidal cloud of points, and any outlying point (representing an unusual residual) should be clearly visible, as shown in Figure 6.18.

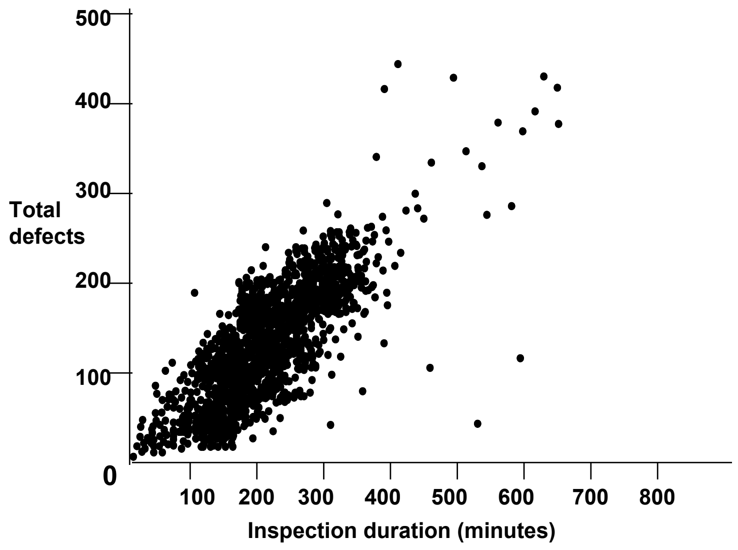


Figure 6.18: A scatter plot of data from a major telecommunications company, with residuals noted

6.3.8 Robust regression

There are a number of robust linear regression approaches, and several statistical textbooks (see [Sprent 2007], for example) describe them. For instance, Theil proposed estimating the slope of the regression line as the median of the slopes of all lines joining pairs of points with different values. For each pair (x_i,y_i) and (x_j,y_j), the slope is

$$b_{ij} = \frac{y_j - y_i}{x_j - x_i}$$

If there are n data points and all the x_i are different, then there are $n(n\text{-}1)/2$ different values for b_{ij}, and b is estimated by the median value. It is also possible to determine the confidence limits for this estimate.

Theil suggested estimating the intercept, a, as the median of all the values

$$a_i = y_i - bx_i$$

Example 6.16: We can apply Theil's method to the relationship between size and effort represented by dataset 1, and we generate the equation

effort(months) = 8.869 + 1.413 size(KLOC)

This equation is quite similar to the regression line obtained by least squares when the atypical value is removed. However, the confidence limits on the estimate of the slope range from -0.22 to 2.13 (at the 0.05 level of significance), which includes 0. Thus, we must conclude that the robust technique implies that the relationship between effort and size is not significant.

6.3.9 Multivariate regression

The regression methods we have considered so far focus on determining a linear relationship between two attributes. Each involves the use of the least squares technique. This technique can be extended to investigate a linear relationship between one dependent variable and two or more independent variables; we call this *multivariate linear regression.* However, we must be cautious when considering a large number of attributes, because:

- It is not always possible to assess the relationship visually, so we cannot easily detect atypical values.
- Relationships among the dependent variables can result in unstable equations. For example, most size and structure attributes are correlated, so it is dangerous to include several size and structure measures in the same multivariate regression analysis.
- Robust multivariate regression methods can be quite complicated.

For multivariate linear regression, we recommend using least squares analysis but avoiding the use of many correlated dependent variables. Be sure to analyze the residuals to check for atypical data points (that is, data points having very large residuals).

6.4. More advanced methods

There are many other statistical methods for analyzing data. In this section, we consider several advanced methods that can be useful in investigating relationships among attributes: classification tree analysis, transformations, and multivariate data analysis.

6.4.1 Classification tree analysis

Many statistical techniques deal with pairs of measures. But often we want to know which measures provide the best information about a particular goal or behavior. That is, we collect data for a large number of measures, and we want to know which ones are the best predictors of the behavior in a given attribute. A statistical technique called *classification tree analysis* can be used to address this problem. This method, applied successfully on data from multiple releases of a large telecommunications system [Khoshgoftaar et al. 2000], allows large sets of metrics data to be analyzed with respect to a particular goal.

Example 6.17: Suppose you have collected data on a large number of code modules, and you want to determine which of the metrics are the best predictors of poor quality. You define poor quality in terms of one of your measures; for instance, you may say that a module is of poor quality if it has more than three faults. Then, a classification tree analysis generates a decision tree to show you which of the other measures are related to poor quality. The tree may look like the one in Figure 6.20, which suggests that if a module is between 100 and 300 lines of code and has a cyclomatic number of at least 15, then it may have a large number of faults. Likewise, if the module is over 300 lines of code, has had no design review, and has been changed at least five times, then it too may have a large number of faults. This technique is useful for shrinking a large number of collected measures to a smaller one, and for suggesting design or coding guidelines. In this example, the tree suggests that, when resources are limited, design reviews may be restricted to large modules, and code reviews may be advisable for large modules that have been changed a great deal.

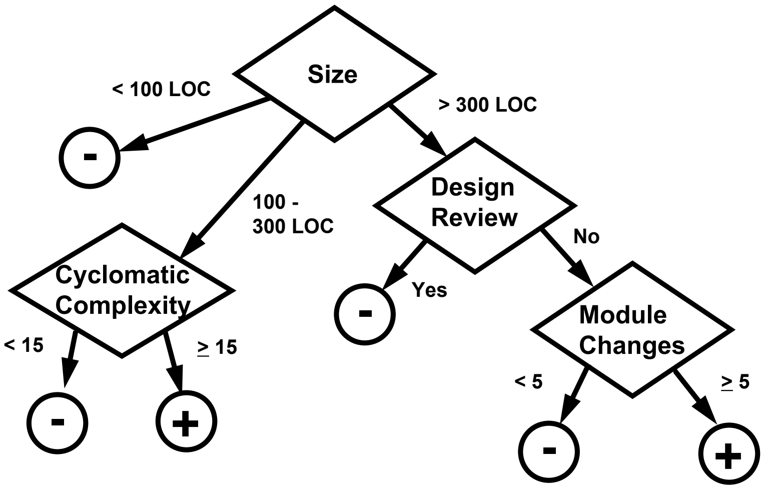


Figure 6.19: Classification tree

6.4.2 Transformations

Sometimes it is difficult to understand the behavior of data in its original form, but much easier if the data are transformed in some way to make the behavior more visible. We noted earlier that it is sometimes possible to transform non-normal data into normal data by applying a logarithmic function. In general, a **transformation** is a mathematical function applied to a measurement. The function *transforms* the original dataset into a new dataset. In particular, if a relationship between two variables appears to be non-linear, it is often convenient to transform one of the attributes in order to make the relationship more linear.

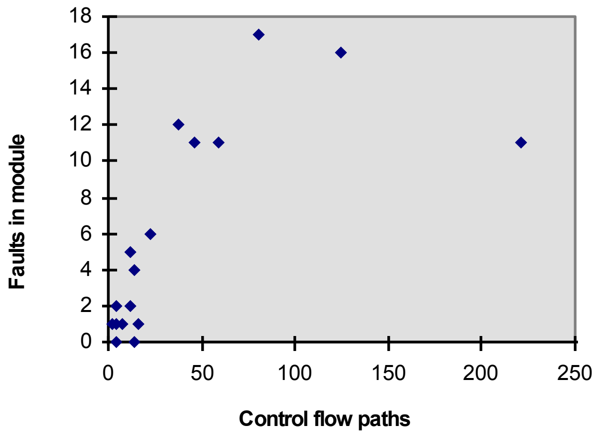


Figure 6.20: Structure graphed against faults, from dataset 2

Example 6.18: Figure 6.20 is a scatter plot of some of the data from dataset 2 in Table 6.1. Here, we have graphed the structure of each module, as measured by control flow paths, against the number of faults found in the module. The relationship between faults and number of paths appears to be non-linear and therefore not suitable for analysis using linear regression.

There are many choices for transformation. Tukey suggested a "ladder" from which to decide on an appropriate transformation function [Mosteller and Tukey 1977]. The ladder consists of a sequence of transformation functions, where the "top" of the ladder is on the left as shown in Figure 6.21.

x^3

x^2

x

$x^{1/2}$

$\log x$

$-x^{-1}$

$-x^{-2}$

$-x^{-3}$

Figure 6.21: Turkey's ladder

We begin by positioning ourselves at the *x* value of the ladder (that is, mid-way on the ladder). If we seem to have a curved rather than linear relationship between two attributes, we use the ladder as follows:

- If the relationship looks positive and convex (as in Figure 6.21), we may transform either the independent variable by going down the ladder (that is, using a square root transformation), or the dependent variable by going up the ladder (that is, using a square transformation).
- If the relationship looks positive and concave, we may transform either the independent variable by going up the ladder, or the dependent variable by going down.

If it is important to obtain an equation for predicting the dependent variable, we recommend transforming the independent variable, so that the dependent variable is in the correct scale.

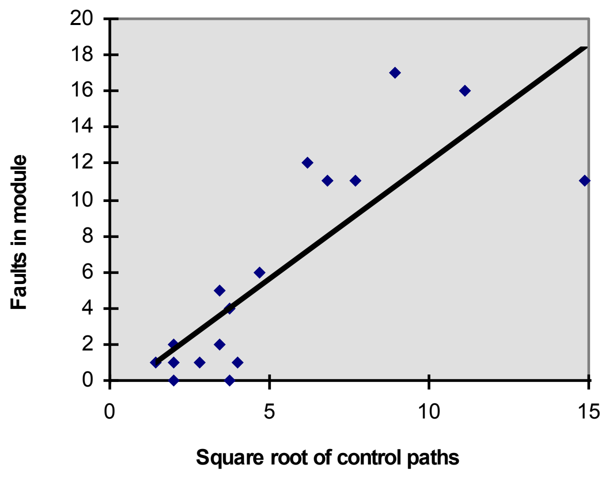


Figure 6.22: Module structure (transformed) plotted against faults

> **Example 6.19:** Figure 6.22 is a graph of the data depicted in Figure 6.20, except that the square root of the control flow paths is plotted, rather than the raw data of dataset 2. The relationship of the transformed data to the module faults is clearly more linear for the transformed data than for the raw data. In fact, the relationship obtained using the raw data accounts for only 46% of the variation of the dependent variable, whereas the relationship obtained using the transformed data accounts for 68% of the variation.

There are many other reasons for transforming data. For example, data are often transformed to cope with relationships of the form

$$y = ax^b$$

where x and y are two attributes, a is a coefficient, and b is an exponent. We can use logarithms to generate a linear relationship of the form

$$\log(y) = \log(a) + b \log(x)$$

Then, $\log(a)$ and b can be estimated by applying the least squares technique to the transformed attribute values. The logarithmic transformation is used frequently to investigate the relationship between project effort and product size, or between project effort and project duration, as we shall see in Chapter 13. However, we must take care in the judgments we make from such transformed data. Plotting relationships on a log-log scale can give a misleading impression of the variability in the raw data. To predict effort (as opposed to log(effort)), you should plot an appropriate line on the *untransformed* scatter plot.

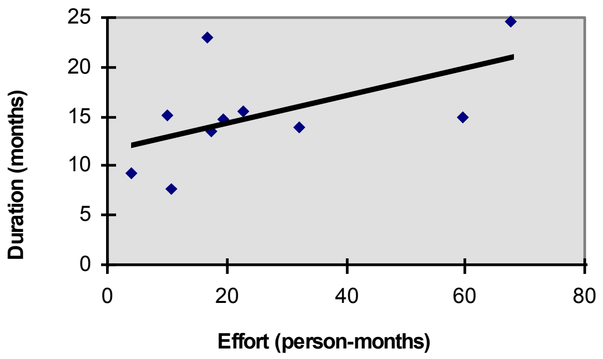


Figure 6.23: Effort plotted against duration, from dataset 1

Example 6.20: The relationship between effort and duration for dataset 1 is depicted in Figure 6.23. A linear relationship is not significant for the raw data. However, when we transform the data using the logarithm of each attribute, the result is the scatter plot of Figure 6.24. This figure shows the linear regression line, which is significant after using the log-log transformation.

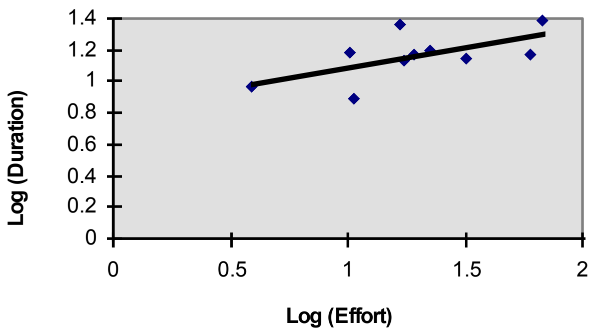


Figure 6.24: Logarithm of effort plotted against logarithm of duration

6.4.3 Multivariate data analysis

There are several different techniques that can be applied to data involving many variables. In this section, we look at three related techniques. *Principal component analysis* can simplify a set of data by removing some of the dependencies among variables. *Cluster analysis* uses the principal components that result, allowing us to group modules or projects according to some criterion. Then, *discriminant analysis* derives an assessment criterion function that distinguishes one set of data from another; for instance, the function can separate data that is "error-prone" from data that is not.

6.4.3.1 Principal component analysis

Often, we measure a large set of attributes where some subset of the attributes consists of data that are related to one another. For example, the size of a project affects the amount of effort required to complete it, so size and effort are related. Suppose we want to predict project duration. If we use both size and effort in a duration prediction equation, we may predict a longer time period than is really required, because we are (in a sense) double counting; the duration suggested by the size measures may also be represented in our effort value. We must take care to ensure that the variables contributing to our equation are as independent of one another as possible.

In general, if we want to investigate the relationship among several attributes, we want to be sure that subsets of related attributes do not present a misleading picture. *Principal component analysis* generates a linear transformation of a set of correlated attributes, such that the transformed variables are independent. In the process, it simplifies and reduces the number of measures with which we must deal. Thus, we begin with n measures x_i, and our analysis allows us to obtain n new variables of the form

$$y_i = a_{i1}x_1 + a_{i2}x_2 + \dots + a_{in}x_n$$

The technique identifies values for the a_{ij} and determines the contribution of y_i to the overall variability of the set of transformed variables.

Although the process creates the same number of transformed variables as original variables, we can reduce the number of variables by examining the *variability* in the relationship. We need fewer transformed variables to describe the variability of the dataset than we had originally, because the attributes that are virtually identical will contribute to a single variable on the new scale. The analysis indicates what proportion of the total variability is explained by each transformation. The transformation accounting for the most variability is called the *first principal component*, the transformation accounting for the next largest amount is called the second principal component, and so on. Principal components that account for less than 5% of the variability are usually ignored.

Example 6.21: During the first phase of the Software Certification Programme in Europe (SCOPE), 39 software metrics related to maintainability were collected from modules produced by five industrial software projects. A principal component analysis reduced the set to six, accounting for nearly 90% of the variation. Size alone explained almost 57% of the variation. In other words, more than half of the variation in the maintainability measure was explained by changes to the size measure. [Neil 1992]

Principal component analysis is available in most statistical packages because it is useful for several purposes:

- To identify the underlying dimensionality of a set of correlated variables.
- To allow a set of correlated variables to be replaced by a set of non-correlated variables in multivariate regression analysis.
- To assist in outlier detection. Each set of attribute values for a software item can be transformed into one or more new variables (that is, the principal components). If the first principal component is plotted against the second, there will be no relationship; however, points that are very distant from the central mass of points are regarded as anomalies.

Table 6.7: Correlations between principal components and the normalized variables for dataset 2 size and structure measures

Original	*Principal Component 1*	*Principal Component 2*	*Principal Component 3*	*Principal Component 4*
LOC	0.81	-0.13	0.57	0.02
Fan-Out	0.92	0.06	-0.28	0.27
Fan-In	-0.10	0.99	0.12	0.03
Path	0.92	0.15	-0.21	-0.28
% variation explained	59.1	24.4	11.5	3.9

Example 6.22: Table 6.7 shows the results of a principal component analysis performed on the four structure and size variables of dataset 2. The variables were normalized prior to performing the analysis (that is, transformations were applied, so that each variable had mean 0 and standard deviation 1). The table shows the correlation between each of the original variables and the transformed variables. It also indicates the percentage of the variability accounted for by each component. The analysis suggests that the four original variables can be represented adequately by the first three transformed variables (that is, the first three principal components). It also shows that the first principal component is related to lines of code, fan-out and paths, since the correlation coefficients are positive and high. Similarly, the second principal component is related to fan-in, and the third principal component is related to lines of code.

6.4.3.2 Cluster analysis

Cluster analysis can be used to assess the similarity of modules in terms of their measurable characteristics. It assumes that modules with similar attributes will evidence similar behavior. First a principal components analysis is performed, producing a reduced set of principal components that explain most of the variation in the behavior being investigated. For example, we saw in Example 6.21 that the behavior, maintainability, was explained by six principal components. Next, cluster analysis specifies the behavior by separating it into two categories, usually exhibiting the behavior and not; in Example 6.21, we can think of the categories as “easy to maintain” or “not easy to maintain.”

There are many cluster algorithms available with statistics packages. These algorithms can be used much as box plots were used for single variables: to cluster the data and identify outliers or unusual cases.

6.4.3.3 Discriminant analysis

Discriminant analysis allows us to separate data into two groups and to determine to which of the two groups a new data point should be assigned. The technique builds on the results of a cluster analysis to separate the two groups. The principal components are used as discriminating variables, helping to indicate to which group a particular module is most likely to belong.

The groups generated by a cluster analysis sometimes have overlap between them. Discriminant analysis reduces this overlap by maximizing the variation between clusters and minimizing the variation within each cluster.

6.5. Multi-criteria decision aids

Most of the techniques we have presented so far are based on classical statistical methods. In this section, we present some newer ways of addressing data analysis, especially when decision-makers must solve problems by taking into account several different points of view. In other words, we have assumed so far that the goal of measurement is clear, but there are times (as we saw in Chapter 3) when some people interpret goals differently from others. We need to be able to analyze data and draw conclusions that address many different interpretations of the same goal, even when the interpretations conflict. Thus, rather than "solving" a problem by finding a universal truth or law exhibited by the data, we need instead to find a subjective problem resolution that is consistent and satisfies all parties involved, even if it is not optimal in the mathematical sense.

To understand why this situation is common in software measurement, consider a software system that is being built to meet requirements about safety or reliability. Data are collected to help answer the question, "Is this software safe?" But the interpretation of "safe" can differ from one person to another, and it is not always clear that we have sufficient evidence to answer the question. Moreover, we often have to balance several considerations. We may ask, "which option provides the most reliability for the least cost?" and we must make decisions that reflect our priorities when there is no clear-cut "best" answer.

Multi-criteria decision aids can help us to address such questions. They draw heavily on methods and results in operations research, measurement theory, probability, fuzzy sets, the theory of social choice, and expert systems. In the 1980s and 1990s, multi-criteria decision aids made great strides, and several computer-based tools have been developed to implement the methods. The book by Vincke provides additional information [Vincke 1992].

In this section, we begin with the basic concepts of multi-criteria decision-making. Then, we examine two classes of methods that have received a great deal of attention: multiple-attribute utility theory (including the analytical hierarchy process), and outranking methods. The latter is far less stringent than the former, and it allows for more realistic assumptions. Our examples are presented at a very high level, so that you can see how the concepts and techniques are applied to software engineering problems. But these techniques are quite complex, and the details are beyond the scope of this book. The end of this chapter suggests other sources for learning the techniques in depth.

6.5.1 Basic concepts of multi-criteria decision-making

We often have a general question to answer, and that question is really composed of a set of more specific decision problems. For example, it is not enough to ask, "How do we build a system that we know is safe?" We must ask an assortment of questions, such as

- Which combination of development methods is most appropriate to develop a safe system in the given environment?

- How much effort should be spent on each of a set of agreed-upon testing techniques in order to assure the safety of this system?
- Which compiler is most appropriate for building a safe system under these conditions?
- What is considered valid evidence of safety, and how do we combine the evidence from different subsets or sources to form a complete picture?
- Which of a set of possible actions should we take after system completion to assess the system safety?

In each case, we have a set of *actions*, *A*, to be explored during the decision procedure. *A* can include objects, decisions, candidates and more. For instance, in the first question above, *A* consists of all combinations of mutually compatible methods selected from some original set. An enumeration of *A* might consist of combinations such as

(Alloy specification, correctness proofs)
(Alloy specification, UML design)
(OCL specification, Z formal verification, proof)
(OOD)
(Specification from Python rapid prototyping, agile development)

and so on. Once we have a set of actions, we define a *criterion*, *g*, to be a function from the set of actions to a totally ordered set, *T*. That *T* is *totally ordered* means that there is a relation *R* on pairs of elements of *T* that satisfies four properties:

1. *R* is *reflexive*: for each element *x* in *T*, the pair (*x*, *x*) is in *R*.
2. *R* is *transitive*: if (*x*, *y*) and (*y*, *z*) are in *R*, then (*x*, *z*) must be in *R*.
3. *R* is *antisymmetric*: if (*x*, *y*) and (*y*, *x*) are in *R*, then *x* must equal *y*.
4. *R* is *strongly complete*: for any *x* and *y* in *T*, either (*x*, *y*) is in *R* or (*y*, *x*) is in *R*.

These conditions guarantee that any action can be compared to any other action using a relation on *T*. We can use the total ordering to compare software engineering techniques, methods and tools.

Example 6.23: Suppose *A* is a set of verification and validation techniques used on a software project. We can define a criterion that maps *A* to the set of real numbers by defining *g* as follows: For each element *a* of *A*,

g(a) = the total effort in person-months devoted to using technique *a*

This measure gives us information about experience with each element of *A*. Alternatively, we can define another criterion, *g'*, to be a mapping to the set of non-negative integers, so that we can compare the effectiveness of two techniques:

g'(a) = the total number of faults discovered when using technique *a*

> We can also have criteria that map to sets that are not numbers. For example, define the set *T* to be {poor, moderate, good, excellent}, representing categories of ease of use. Then we can define a criterion that maps each element of *A* to an element of *T* that represents the ease of use for a particular technique, as rated subjectively by an expert. In this example, *T* is totally ordered, since
>
> poor < moderate < good < excellent

Suppose we consider a family of criteria defined on a set of actions, *A*. If the family is *consistent* (in a way that is described more fully by Roy [1990]), then a multicriteria decision problem can be any one of the following:

- Determine a subset of *A* considered to be best with respect to the family of criteria (the *choice problem*).
- Divide *A* into subsets according to some norms (the *sorting problem*).
- Rank the actions of *A* from best to worst (the *ranking problem*).

To solve these problems, a decision-maker must compare each pair actions, *a* and *b*, in one of three ways:

1. Strict preference for one of the actions.
2. Indifference to either action.
3. Refusal or inability to compare the actions.

Each of these choices defines a relation between *a* and *b*. The set of all of these relations forms a *preference structure* on *A*. We can also define a *preference relation S* using only conditions 1 and 2, where *a* is related to *b* if and only if either *a* is strictly preferred to *b*, or there is indifference between the two. A classic problem of decision optimization is to define a numerical function that preserves *S*. In other words, we want to define a function, *f*, from the set *A* to some number system, *N*, that satisfies both of these conditions:

$$f(a) > f(b) \text{ if and only if } a \text{ is strictly preferred to } b$$
$$f(a) = f(b) \text{ if and only if there is indifference between } a \text{ and } b$$

This function may look familiar to you, as it is a mathematical way of describing the representation condition, introduced in Chapter 2. In multi-criteria decision-making, a key problem is optimizing *f*. Here, "optimization" means that f must satisfy the stated conditions in a way that optimizes one or more attributes of the elements of *A*, such as cost or quality. When *N* is the set of real numbers, then no such function *f* exists if there are two actions that are incomparable. Unfortunately, there are many real-world situations where incomparabilities exist; we can include the incomparabilities by mapping to other types of number systems, such as vectors of real numbers, as we did in Chapter 2.

When we can preserve the preference structure by mapping to the real numbers, then we can rank the actions from best to worst, with possible ties when there is indifference between two actions; this mapping is called the *traditional model*. Such a relation is called a *complete preorder*. If there are no ties, then the preference structure is a *complete order*. Any criterion for which the underlying preference structure is a complete preorder is called a *true criterion*.

In the traditional model, indifference must be transitive; that is, if there is indifference between *a* and *b*, and indifference between *b* and *c,* then there must be indifference between *a* and *c*. However, sometimes this condition is not realistic, as when there are "sensibility thresholds" below which we are indifferent; multi-criteria decision-making can be extended to cover this case, but here we focus on the traditional model and assume transitivity.

Suppose *a* and *b* are possible actions, and we have a set of *n* criteria $\{g_i\}$. We say that *a dominates b* if $g_i(a) \geq g_i(b)$ for each *i* from 1 to *n*. An action is said to be *efficient* if it is strictly dominated by no other action.

Example 6.24: We are considering eight software packages, P1 through P8, to select one for use in a safety-critical application. All of the packages have the same functionality. There are four criteria that govern the selection: cost (measured in dollars), speed (measured in number of calculations per minute on a standardized set of test data), accuracy and ease of use. The latter two criteria are measured on an ordinal scale of integers from 0 to 3, where 0 represents *poor*, 1 represents *fair*, 2 represents *good*, and 3 represents *very good*. Table 6.8 contains the results of the ratings for each criterion. To ensure that the dominance relation holds, we have changed the sign of the values for cost. In this example, we see that

P2 dominates P1
P5 dominates both P4 and P6
P3, P7 and P8 dominate no other package
P2, P5, P3, P7 and P8 are efficient

Table 6.8: Ratings for each software package against four criteria

Software package	g_1*: -Cost*	g_2*: Speed*	g_3*: Accuracy*	g_4*: Ease of use*
P1	-1300	3000	3	1
P2	-1200	3000	3	2
P3	-1150	3000	2	2
P4	-1000	2000	2	0
P5	-950	2000	2	1
P6	-950	1000	2	0
P7	-900	2000	1	0
P8	-900	1000	1	1

The first task in tackling a multi-criteria decision problem is reducing the set of actions to a (probably smaller) subset of efficient actions. We may find that there is only one efficient action, in which case that action is the simple solution to our problem. However, more often the dominance relation is weak, and multi-criteria decision activities involve enriching the dominance relation by considering all relevant information.

To do this, we consider a vector formed by evaluating all of the criteria for a given action. That is, for each action, a, we form a vector $<g_1(a), g_2(a), ..., g_n(a)>$. Then the collection of all of these n-tuples is called the *image* of A. In Example 6.24, the image of A is the set of eight 4-tuples that correspond to the rows in Table 6.8. For each i from 1 to n, we can identify the (not necessarily unique) action a_i^* in A that is best according to the criterion g_i. The *ideal point* is the point $(z_1, z_2, ..., z_n)$, where $z_i = g_i(a_i^*)$. For instance, the ideal point in Example 6.24 is (-900, 3000, 3, 2).

We can apply all of the criteria to each ideal point, so that we compute all possible $G_{ij} = g_j(a_i^*)$. The n by n matrix formed by the G_{ij} is called the *payoff matrix*; this matrix is unique only if each criterion achieves its maximum at only one action. The diagonal of a payoff matrix is the ideal point.

Example 6.25: There are two different payoff matrices for the actions and criteria in Example 6.24:

$$M = \begin{bmatrix} a_1^* = P8 & -900 & 1000 & 1 & 1 \\ a_2^* = P2 & -1200 & 3000 & 3 & 2 \\ a_3^* = P2 & -1200 & 3000 & 3 & 2 \\ a_4^* = P3 & -1150 & 3000 & 2 & 2 \end{bmatrix}$$

$$M' = \begin{bmatrix} a_1^* = P7 & -900 & 2000 & 1 & 0 \\ a_2^* = P3 & -1150 & 3000 & 2 & 2 \\ a_3^* = P1 & -1300 & 3000 & 3 & 1 \\ a_4^* = P2 & -1200 & 3000 & 3 & 2 \end{bmatrix}$$

For a given payoff matrix, the *nadir* is the point whose ith coordinate is the minimum of the values in the ith column. In Example 6.25, the nadir for matrix M is (-1200, 1000, 1, 1); for M', the nadir is (-1300, 2000, 1, 0).

These concepts can be used to help solve the multi-criteria decision problem. For instance, it can be shown that a positive linear combination of criteria always yields an efficient action, and that efficient actions can be characterized as those that minimized a certain distance (called the *Tchebychev distance*) to a point that slightly dominates the ideal point. Parametric optimization techniques can then be used to determine efficient actions.

However, there are often more constraints on the problem than these. For instance, in Table 6.8, we see that, although P3 is $50 more expensive, its accuracy is greater than that of P2. Our decision-makers may believe that it is worth paying the extra money to increase the accuracy. Such observations are called *substitution rates of criteria*; they allow us to add a particular amount to one criterion to compensate for a loss of one unit of another criterion. Although the assignment of substitution rates is subjective, the technique allows us to control the insertion of subjectivity, and to be consistent in applying our subjective judgments.

Suppose that, because of the safety-criticality of the application, the decision-makers prefer to pay a lot more for a package to get the highest degree of accuracy, no matter what the speed or ease of use. In this situation, two of the criteria (namely, speed and accuracy) are said to be *preferentially independent*.

6.5.2 Multi-attribute utility theory

Multi-attribute utility theory (MAUT) takes constraints such as a preference for accuracy over costs into account, providing another approach to organize subjective judgments for use in choosing between multiple alternatives. MAUT assumes that a decision-maker want to maximize a function of various criteria. Two types of problems present themselves:

- *The representation problem*: What properties must be satisfied by the decision-maker's preferences so that they can be represented by a function with a prescribed analytical form?
- *The construction problem*: How can the maximization function be constructed, and how can we estimate its parameters?

Example 6.26: Suppose we are given four criteria for assessing which verification and validation techniques should be used on a software module; these criteria are described in Table 6.9.

Table 6.9: Criteria for assessing verification and validation techniques

Criterion	*Measurement scale*
g_1: Effort required	{little, moderate, considerable, excessive}
g_2: Coverage	{bad, reasonable, good, excellent}
g_3: Tool support	{no, yes}
g_4: Ranking of usefulness by expert	{1, 2, ..., n} where n is number of techniques

A decision-maker rates four techniques, and the results are shown in Table 6.10. We have several choices for defining a utility function, U, which is a sum of functions U_i on each g_i. One possibility is to define each U_i to be a transformation onto the unit interval, [0, 1]. For instance, we may define:

U_1(little) = 0.8 U_1(moderate) = 0.5 U_1(considerable)= 0.2 U_1(excessive) = 0

U_2(bad) = 0 U_2(reasonable) = 0.1 U_2(good) = 0.3 U_2(excellent) = 0.6
U_3(no) = 0.2 U_3(yes) = 0.7
$U_4(x) = 1/x$

Thus, for any technique, we rate the availability of tool support as having greater utility than excellent coverage, and we rate the greatest utility to be ranked top by an expert. The final result can be evaluated by summing the functions for each technique:

U(inspections) = 0.2 + 0.6 + 0.2 + 1 = 2
U(proof) = 0 + 0.3 + 0.2 + 0.5 = 1
U(static analysis) = 0.5 + 0 + 0.7 + 0.25 = 1.45
U(black box) = 0.8 + 0.3 + 0.2 + 0.33

Table 6.10: Evaluation of four verification and validation techniques

Technique	g_1	g_2	g_3	g_4
Code inspection	considerable	excellent	no	1
Formal proof	excessive	good	no	2
Static analysis	moderate	bad	yes	4
Black box testing	little	good	no	3

It is easy to see that the criteria in Table 6.9 are non-orthogonal, as the criteria are not independent. In an ideal situation, the criteria should be recast into an orthogonal set, so that the relative importance of each is clear and sensible. However, in many situations, it is not possible or practical to derive an orthogonal set. Nevertheless, MAUT helps us to impose order and consistency to the analysis process. Sometimes, the act of applying MAUT makes the decision-maker more aware of the need for orthogonality, and the criteria are changed. Otherwise, MAUT helps to generate a reasonable decision when the "best" is not possible.

The analytic hierarchy process (AHP) is a popular MAUT technique [Saaty and Vargas 2001]. AHP begins by representing the decision problem as a hierarchy graph, where the top node is the main problem objective and the bottom nodes represent actions. At each level of the hierarchy, a decision-maker makes a pairwise comparison of the nodes from the viewpoint of their contribution to each of the higher-level nodes to which they are linked. The pairwise comparison involves *preference ratios* (for actions) or *importance ratios* (for criteria), evaluated on a particular numerical scale. Then, the "value" of each node is calculated, based on computing the eigenvalues of the matrix of pairwise comparisons. Each action's global contribution to the main objective is calculated by aggregating the weighted average type. A software package called Expert Choice supports use of AHP.[2]

AHP has been used in dependability assessment [Auer 1994], and Vaisanene and colleagues have applied it to a safety assessment of programmable logic components.

[2] www.expertchoice.com

[Vaisanene *et al.* 1994] AHP was also used to help NASA choose from among several strategies for improving a safety feature of the Space Shuttle. [Frank 1995]

6.5.3 Outranking methods

MAUT can be very valuable, resulting in a ranking of actions from best to worst. However, such rankings are based on assumptions that are unrealistic in many situations. For many problems, we do not a complete ranking, so we can do without the unrealistic assumptions. However, the dominance relation alone is generally too weak to be useful. Outranking methods seek to enrich the dominance relation without having to make the assumptions of MAUT.

An *outranking relation* is a binary relation on a set of actions defined in the following way. Consider two actions *a* and *b*. Given what is known about the decision-maker's preferences, the quality of the valuations of the actions, and the nature of the problem, if there are enough arguments to decide that *a* is at least as good as *b* and no essential reason to refute that statement, then we say that *a* is related to *b*. This definition is informal, and published outranking methods differ in the way that they are formalized. No matter the definition, the outranking relation is neither necessarily complete nor transitive. [Roy 1990]

Any outranking method has two steps:

1. Building the outranking relation
2. Exploiting the relation with regard to the chosen problem statement

To give you an idea of how outranking works, we present a brief example, using the Electre I method [Figueira et al. 2005].

Example 6.27: Table 6.11 lists four criteria for assessing combinations of verification and validation techniques. For instance, action 1 might involve formal proof followed by code inspection, while action 2 might be formal proof plus static code analysis. Each criterion must map onto a totally ordered set, and each is assigned a weight, indicated by the number in parentheses. (For example, the weight of the first criterion is 5.)

Table 6.11: Criteria for assessing combined verification and validation techniques

Action	*g_1: Effort required (weight = 5)*	*g_2: Potential for detecting critical faults (weight = 4)*	*g_3: Coverage achieved (weight = 3)*	*g_4: Tool support (weight = 3)*
1	excessive	excellent	good	yes
2	considerable	excellent	average	yes
3	considerable	good	good	yes
4	moderate	good	good	no
5	moderate	good	average	yes

6	moderate	reasonable	good	yes
7	little	reasonable	average	no

For each ordered pair of actions (*a, b*), we compute a *concordance index* by adding the weights of all criteria g_i for which $g_i(a)$ is greater than $g_i(b)$. Thus, for the pair (1, 2), action 1 is at least as good as action 2 with respect to all but the first criterion, so we sum the weights (4 + 3 + 3) to yield a concordance index of 10. The full set of concordance indices is shown in Table 6.12. In a sense, this table is the "first draft" of the preference structure (that is, the outranking relation). We say that *a* is preferable to *b* when the concordance index for (*a, b*) is larger than for (*b, a*).

Table 6.12: Concordance indices

	1	*2*	*3*	*4*	*5*	*6*	*7*
1	-	10	10	10	10	10	10
2	12	-	12	7	10	7	10
3	11	11	-	11	10	10	10
4	8	8	12	-	12	12	10
5	8	11	12	12	-	12	10
6	11	11	11	11	11	-	10
7	5	8	5	8	8	9	-

Next, we restrict the structure by defining a *concordance threshold*, *t*, so that *a* is preferred to *b* only if the concordance index for (*a, b*) is at least as large as *t*. Suppose *t* is 12. Then, according to Table 6.12, action 2 is still preferable to action 1, but we no longer have preference of action 1 to action 7 because the concordance index for (1, 7) is not large enough.

We further refine the preference structure to include other constraints. For instance, suppose that for criterion g_1 (effort required) we never allow action *a* to outrank action *b* if $g_1(a)$ is excessive and $g_1(b)$ is little. In other words, regardless of the values of the other criteria, *b* is so superior to *a* with respect to g_1 that we veto it being outranked by *a*. In general, we handle this situation by defining a *discordance set* for each criterion, containing the ordered pairs of values for the criterion for which the outranking is refused.

For example, let us define

D_1 = {(excessive, little), (considerable, little)}
D_2 = {(moderate, excellent)}
$D_3 = D_4$ = {}

Then we can define the outranking relation by saying that one action outranks another provided that its concordance index is at least as large as the threshold, and provided that for each criterion, the preference is not vetoed by the discordance set. The full outranking relation for the example is shown in Figure 6.25.

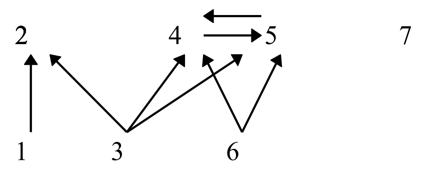


Figure 6.25: Graph of outranking relation. An arc from node *x* to node *y* indicates that *y* outranks *x*.

Finally, we exploit this relationship to find a subset of actions that is optimal with respect to the outranking. In graph theory, such a set (of nodes) is called a *kernel.* There are several graph-theoretic techniques for determining kernels, thus enabling us to find the best course of action, given our constraints. From Figure 6.25, we can see that there are three kernels, namely {2,4,7} and {2,5,7}; the actions 4 and 5 are considered to be tied.

6.5.4 Bayesian evaluation of multiple hypotheses

You may have noticed that AHP, MAUT, and Outranking Methods depend on operations on data that, according to Chapter 2 represent inadmissible transformations. Many of the attributes and attribute weights appear to be rankings and thus are ordinal scale measures. Yet, the methods perform multiplication and/or addition on these values. It would be better if we could use an analysis using only well-defined and meaningful transformations.

The Bayesian approach, based on a theory introduced by Thomas Bayes in 1763, analyzes multiple variables in terms of three key factors: the probability of the occurrence of a set of events, and the outcome of each event should it occur, and the dependencies between events. The probabilities and outcomes are expressed in terms of ratio scales measures making the analyses meaningful. The Bayesian approach supports the use of qualitative beliefs concerning probabilities and outcomes, and is dynamic – it supports the updating beliefs in response to new evidence. In addition, unlike classical data analysis techniques described in Section 6.2, the Bayesian approach focuses on the magnitude of relations rather than their precision.

This elegant approach has been widely used in law, medicine, finance, and engineering. Tools that support Bayesian analyses are now readily available. This approach is well suited to analyzing software engineering problems. Thus, we devote Chapter 7 to an in-depth description of the Bayesian approach applied to software engineering problems.

6.6. Overview of statistical tests

This chapter has mentioned several statistical tests that can be used to analyze your data. We have presented techniques in terms of the type of relationship we seek (linear, multivariate, and so on), but our choice of technique must also take into account the number of groups being compared, the size of the sample, and more. In this section, we describe a few statistical tests, so that you can see how each one is oriented to a particular experimental situation. The section is organized in terms of sample and experiment type, as summarized in Table 6.13.

Table 6.13: Examples of statistical tests

Sample type	***Nominal***	***Ordinal***	***Interval***	***Ratio***
One sample	Binomial. Chi-square	Kolmogorov-Smirnov One-sample runs Change point		Single-sam *T*-test
Two related samples	McNemar change	Sign test Wilcoxon signed ranks	Permutation	Matched gr T-test
Two independent samples	Fisher exact Chi-square	Median test Wilcoxon-Mann-Whitney Robust rank order Kolmogorov-Smirnov 2-sample Siegel-Tukey	Permutation	Population
K related samples	Cochran *Q*-test	Friedman 2-way ANOVA Page test (ordered alternatives)		Within-gro ANOVA
K independent samples	Chi-squared	Kruskal-Wallis 1-way ANOVA Jonckheere test		Between-gr ANOVA

6.6.1 One-group tests

In a one-group design, measurements from one group of subjects are compared with an expected population distribution of values. These tests are appropriate when an experimenter has a single set of data with an explicit null hypothesis concerning the value of the population mean. For example, you may have sampled the size of each of a group of modules, and you hypothesize that the average module size is 200 lines of code.

For normal distributions, the parametric test is called the *t*-test (or, equivalently, the Student's *t*-test, single sample *t*-test, or one-group *t*-test). Because the mean of the data is involved, the test is appropriate only for data that are on the interval scale or above.

There are several alternatives for non-parametric data.

6.6.1.1 Binomial test

The binomial test should be used when

1. the dependent variable can take only two distinct, mutually-exclusive and exhaustive values (such as “module has been inspected” and “module has not been inspected”), and
2. the measurement trials in the experiment are independent.

6.6.1.2 Chi-squared test for goodness of fit

The chi-squared test is appropriate when

1. the dependent variable can take two or more distinct, mutually-exclusive and exhaustive values (such as “module has fewer than 100 lines of code,” “module has between 100 and 300 lines of code” and “module has more than 300 lines of code”),
2. the measurement trials in the experiment are independent, and
3. none of the categories has an observed or expected frequency of less than five occurrences.

6.6.1.3 Kolmogorov-Smirnov one-sample test

This test assumes that the dependent measure is a continuous, rather than discrete, variable. It assesses the similarity between an observed and an expected cumulative frequency distribution.

6.6.1.4 One-sample runs test

This test determines if the results of a measurement process follow a consistent sequence, called a *run*. For example, it can be used in analyzing the reasons for system downtime, to determine if one particular type of hardware or software problem is responsible for consecutive system crashes. The test assumes that the successive measurement trials are independent.

6.6.1.5 Change-point test

This test determines whether the distribution of some sequence of events has changed in some way. It assumes that the observations form an ordered sequence. If, at some point in the distribution of observed measures, a shift in the median (or middle) score has occurred, the test identifies the change point.

6.6.2 Two-group tests

Single-sample designs and tests are often used to decide if the results of some process are straying from an already-known value. However, these designs are not appropriate when an experimentally established comparison is required.

Two-group designs allow you to compare two samples of dependent measures drawn from related or matched samples of subjects (as in a within-subjects design), or from two independent groups of subjects (as in a between-subjects design). The appropriate statistical test depends on whether the samples are independent or related in some way.

6.6.2.1 Tests to compare two matched or related groups

The parametric *t*-test for matched groups applies when the dependent measurement is taken under two different conditions, and when one of the following additional conditions has been met:

1. The same subject is tested in both conditions (that is, a within-subjects or repeated-measures design has been used).
2. Subjects are matched according to some criterion (that is, a matched-groups design has been used).
3. Pre-screening has been used to form randomized blocks of subjects (that is, a randomized block design has been used).

This test assumes the following:

- Subjects have been randomly selected from the population. When different subjects have been used in each condition, assignment to the conditions should be random.
- If fewer than 30 pairs of dependent measures are available, the distribution of the differences between the two scores should be approximately normal.

Like the single-sample *t*-test, this test also requires measurement data to be at least on the interval scale.

Non-parametric alternatives to this test include the McNemar change test, the sign test, and the Wilcoxon signed ranks test. The *McNemar change test* is useful for assessing change on a dichotomous variable, after an experimental treatment has been administered to a subject. For example, it can be used to assess whether programmers switch preferences for language type after receiving training in procedural and object-oriented techniques. The test assumes that the data are frequencies that can be classified in dichotomous terms.

The *sign test* is applied to related samples when you want to establish that one condition is greater than another on the dependent measure. The test assumes that the dependent measure reflects a continuous variable (such as experience) rather than categories. For example, you may use this test to determine whether a particular type of programming construct is significantly easier to implement on a given system. The *Wilcoxon signed ranks test* is similar to the sign test, taking into account the magnitude as well as the direction of difference; it gives more weight to a pair that shows a large difference than to a pair that shows a small difference.

6.6.2.2 Tests to compare two independent groups

The parametric test appropriate for independent groups or between-subjects designs is the *t*-test for differences between population means. This test is applied when there are two groups of different subjects that are not paired or matched in any way. The test assumes the following:

1. Subjects have been randomly selected from the population and assigned to one of the two treatment conditions
2. If fewer than 30 subjects are measured in each group, the distribution of the differences between the two scores should be approximately normal, and
3. The variance, or spread, of scores in the two population groups should be equal or homogeneous.

As before, the data must be measured on at least an interval scale to be suitable for this test.

Several non-parametric alternatives to this test are available; they are described in most statistics books.

6.6.3 Comparisons involving more than two groups

Suppose you have measured an attribute of *k* groups, where *k* is greater than two. For example, you are comparing the productivity rates of ten projects, or you want to look at a structural attribute for each of 50 modules. Here, the appropriate statistical analysis technique is an *analysis of variance* (ANOVA). This class of parametric techniques is suitable for data from between-subjects designs, within-subjects designs, and designs that involve a mixture of between- and within-subjects treatments. Complementary non-parametric tests are available but are beyond the scope of this book; some are listed in Table 6.13.

ANOVA results tell you whether at least one statistically significant difference exists somewhere within the comparisons drawn by the analysis. Statistical significance in an ANOVA is reflected in the *F* statistic that is calculated by the procedure. When an *F* statistic is significant, you then apply multiple comparisons tests to determine which levels of a factor differ significantly from the others.

As we noted in Chapter 4, when your experimental design is complex, it is best to consult a statistician to determine which analysis techniques are most appropriate. Because the techniques require different types of data and control, it is sensible to lay out your analysis plans and methods *before* you begin your investigation; otherwise, you risk collecting data that is insufficient or inappropriate for the analysis you need to do.

6.7. Summary

Datasets of software attribute values must be analyzed with care, because software measures are not usually normally distributed. We have presented several techniques here that address a wide variety of situations: differing data distributions, varying measurement scales, varying sample sizes, and differing goals. In general, it is advisable to:

- Describe a set of attribute values using box plot statistics (based on median and quartiles) rather than on mean and variance.
- Inspect a scatter plot visually when investigating the relationship between two variables.
- Use robust correlation coefficients to confirm whether or not a relationship exists between two attributes.
- Use robust regression in the presence of atypical values to identify a linear relationship between two attributes, or remove the atypical values before analysis.
- Always check the residuals by plotting them against the dependent variable.
- Use Tukey's ladder to assist in the selection of transformations when faced with non-linear relationships.
- Use principal component analysis to investigate the dimensionality of datasets with large numbers of correlated attributes.

It is also helpful to consider using more advanced techniques, such as multi-attribute utility theory, to help you choose a "good" solution, rather than the "best" solution, subject to the constraints on your problem. But most important, we have demonstrated that your choice of analysis technique must be governed by the goals of your investigation, so that you can support or refute the hypothesis you are testing.

6.8. Exercises

1. Why are statistics necessary? Why can't experimenters just look at the data and decide for themselves what is important? On the other hand, which of the following is more important? a) the results of a statistical test, or; b) the results of the intra-ocular significance test (that is, when the effect leaps out and hits you between the eyes)?
2. Can a comparison that does not achieve statistical significance still be important?
3. Three years ago the software development manager introduced changes to the development practices at your company. These changes were supposed to ensure that more time was spent up-front on projects, rather than coding Suppose you have the following data giving actual effort by software development phase for the last 5 projects (in chronological order). Provide an appropriate graphical representation for the manager so that she can see whether the changes have had an effect.

	Project 1	**Project 2**	**Project 3**	**Project 4**	**Project 5**
Requirements	120	100	370	80	410
Specification	320	240	490	140	540
High level design	30	40	90	40	60

Detailed design	170	190	420	120	340
Coding	1010	420	1130	250	1200
Testing	460	300	580	90	550

4. Construct a box plot for the paths measure in dataset 2 of Table 6.1.
5. Draw the scatter plot of the LOC and the path measures for the modules in dataset 2 of Table 6.1. Are there any unusual values?
6. Obtain the Spearman rank correlation coefficient and the Pearson correlation coefficient for paths and LOC for dataset 2 of Table 6.1. Why do you think the rank correlation is larger than the Pearson correlation? (Hint: look at the scatter plot produced for Exercise 5)
7. Using a statistical package or spreadsheet, calculate the least squares regression line for faults against lines of code for all the modules in dataset 2 of Table 6.1. Calculate the residuals and plot the residual against the dependent variable (lines of code). Can you identify any outlying points?
8. If you have access to a statistical package, do a principal component analysis of the normalized LOC, fan-out, fan-in, and paths measures from dataset 2 of Table 6.1. Produce a scatter plot of the first principal component against the second and identify the atypical values.
9. What result do you think you would get if you did a principal component analysis on the raw rather than the normalized size and structure measures in dataset 2 of Table 6.1? If you have a statistical package, perform a principal component analysis on the raw size and structure data and see if the results confirm your opinion.
10. Company X runs a large software system S that has been developed in-house over a number of years. The company collects information about software defects discovered by users of S. During the regular maintenance cycle, each defect is traced to one of the 9 subsystems of S (labeled with letters A through I), each of which is the responsibility of a different team of programmers. The table below summarizes information about new defects discovered during the current year:

System	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*
Defects	35	0	95	35	55	40	55	40	45
Size (KLOC)	40	100	5	50	120	70	60	100	40

Suppose you are the manager of system S.

i. Compute the defect density for each subsystem.
ii. Use simple outlier analysis to identify unusual features of the system.
iii. What conclusions can you draw from your outlier analysis?
iv. What are the basic weaknesses of the metrics data as currently collected?
v. What simple additions or changes to the data-collection strategy would significantly improve your diagnostic capability?

11. The table below contains three measures for each of seventeen software modules. LOC is the number of lines of code in the module, CFP the number of control flow paths in the module, and Faults the number of faults found in the module. Construct a box plot for each of the three measures, and identify any outliers. What conclusions can you draw? What recommendations can you make to address the problems you have discovered?

Module	LOC	CFP	Faults
A	15	4	0
B	28	6	15
C	40	2	10
D	60	26	1
E	60	14	0
F	95	18	15
G	110	12	9
H	140	12	6
I	170	54	6
J	180	36	20
K	185	28	14
L	190	32	20
M	210	54	15
N	210	46	18
P	270	128	58
Q	400	84	59
R	420	120	43

12. The table below contains metrics from 17 software systems under investigation: Total number of thousands lines of code for the system (KLOC), average module size (MOD) measured in lines of code, and total number of faults found per thousand lines of code (FD). Construct a box plot for each metric, and identify any outliers.

System	KLOC	MOD	FD
A	10	15	36
B	23	43	22
C	26	61	15
D	31	10	33
E	31	43	15
F	40	57	13
G	47	58	22

H	52	65	16
I	54	50	15
J	67	60	18
K	70	50	10
L	75	96	34
M	83	51	16
N	83	61	18
P	100	32	12
Q	110	78	20
R	200	48	21

6.9. Further Reading

There are several good books on statistics and their application.

P. G. Hoel, *Introduction to Mathematical Statistics*, 5th edition, John Wiley and Sons, New York, 1984.

Roland Caulcutt, *Statistics in Research and Development Second Edition*, Chapman and Hall, London, England, 1991.

Christopher Chatfield, *Statistics for Technology: A Course in Applied Statistics Third Edition (Revised)*, Chapman and Hall, London, England, 1998.

R. Lyman Ott and Micheal T Longnecker, *An Introduction to Statistical Methods and Data Analysis 6th Edition,* Duxbury Press, 2008.

The book by Siegel and Castellan is a classic text on nonparametric statistics.

S. Siegel and N. J. Castellan, Jr., *Nonparametric Statistics for the Behavioral Sciences*, 2nd edition, McGraw-Hill, New York, 1988.

For descriptions of box plots and other exploratory data analysis, consult Hoaglin, Mosteller and Tukey.

D. C. Hoaglin, F. Mosteller and J. W. Tukey, *Understanding Robust and Exploratory Data Analysis*, John Wiley and Sons, New York, 2000.

Sometimes, you have two datasets representing the same situation. Each is different from the other, in terms of distribution, mean or variance, but both were collected with reasonable care. This problem arises frequently but is rarely discussed in standard statistical texts; Barford explains how to tell which is the better dataset for answering your questions.

Barford, N. C., *Experimental Measurements: Precision, Error and Truth Second Edition*, Addison-Wesley, Reading, Massachusetts, 1985.

For an excellent overview of MCDA, you should consult:

Vincke P, *Multicriteria Decision Aid*, John Wiley and Sons, New York, 1992.

Additional references on multi-attribute analysis methods are:

Roy B., “Decision aid and decision making,” *European Journal of Operational Research*, volume 45, pp. 324-331, 1990.

Thomas L. Saaty and Luis G. Vargas, *Models, methods, concepts & applications of the analytic hierarchy process*, Kluwer Academic Publishers, 2001.