Exploratory factor analysis

17

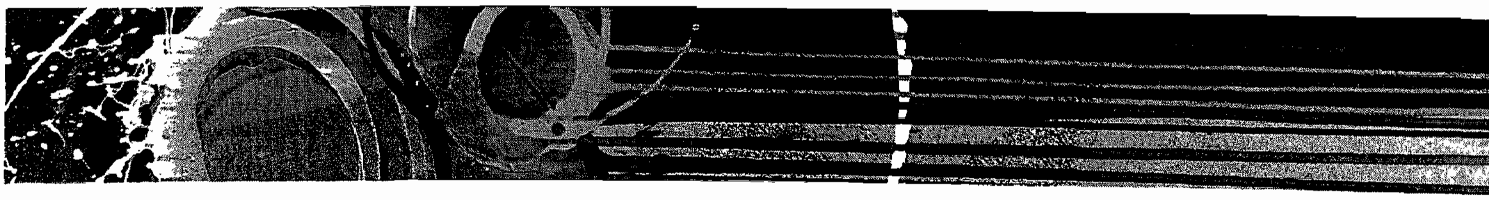


FIGURE 17.1 Me at Niagara Falls in 1998. I was in the middle of writing the first edition of the SPSS version of this book at the time. Note how fresh-faced I look

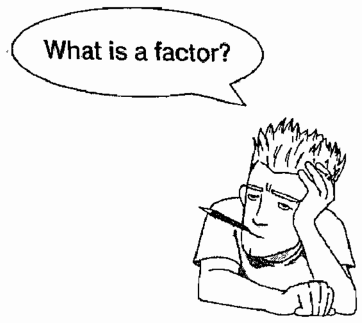
17.1. What will this chapter tell me? ①

I was a year or so into my Ph.D., and, thanks to my initial terrible teaching experiences, I had developed a bit of an obsession with over-preparing for classes. I wrote detailed handouts and started using funny examples. Through my girlfriend at the time I met Dan Wright (a psychologist, who was in my department but sadly moved to Florida). He had published a statistics book of his own and was helping his publishers to sign up new authors. On the basis that my handouts were quirky and that I was too young to realize that writing a textbook at the age of 23 was academic suicide (really, textbooks take a long time to write and they are not at all valued compared to research articles), I was duly signed up. The commissioning editor was a man constantly on the verge of spontaneously combusting with intellectual energy. He can start a philosophical debate about literally anything: should he ever be trapped in a elevator he will be compelled to attempt to penetrate the occupants' minds with probing arguments that the elevator doesn't exist, that they don't exist, and that their entrapment is an illusory construct generated by their erroneous beliefs in the physical world. Ultimately, though, he'd still be a man trapped in an elevator (with several exhausted corpses). A combination of his unfaltering self-confidence, my fear of social interactions with people I don't know, and my utter bemusement that anyone would want me to write a book made me incapable of saying anything sensible to him. Ever. He must have thought that he had signed up an imbecile. He was probably right. (I find him less intimidating since thinking up the elevator scenario.) The trouble with agreeing to write books is that you then have to write them. For the next two years or so I found myself trying to juggle my research, a lectureship at the University of London, and writing a book. Had I been writing a book on heavy metal it would have been fine because all of the information was moshing away in my memory waiting to stage-dive out. Sadly, however, I had agreed to write a book on something that I new nothing about: statistics. I soon discovered that writing the book was like doing a factor analysis: in factor analysis we take a lot of information (variables) and the R program effortlessly reduces this mass of confusion into a simple message (fewer variables) that is easier to digest. The program does this (sort of) by filtering out the bits of the information overload that we don't need to know about. It takes a few seconds. Similarly, my younger self took a mass of information about statistics that I didn't understand and filtered it down into a simple message that I *could* understand: I became a living, breathing factor analysis ... except that, unlike R, it took me two years and some considerable effort.

17.2. When to use factor analysis ②

In the social sciences we are often trying to measure things that cannot directly be measured (so-called **latent variables**). For example, management researchers (or psychologists even) might be interested in measuring 'burnout', which is when someone who has been working very hard on a project (a book, for example) for a prolonged period of time suddenly finds themselves devoid of motivation, inspiration, and wants to repeatedly head-butt their computer screaming 'please Mike, unlock the door, let me out of the basement, I need to feel the soft warmth of sunlight on my skin!'. You can't measure burnout directly: it has many facets. However, you can measure different aspects of burnout: you could get some idea of motivation, stress levels, whether the person has any new ideas and so on. Having done this, it would be helpful to know whether these differences really do reflect a single variable. Put another way, are these different variables driven by the same underlying variable? This chapter will look at factor analysis (and principal components analysis) – a technique for identifying groups or clusters of variables. This technique has three main uses: (1) to understand the structure of a set of variables (e.g., pioneers of intelligence such as Spearman and Thurstone used factor analysis to try to understand the structure of the latent variable 'intelligence'); (2) to construct a questionnaire to measure an underlying variable (e.g., you might design a questionnaire to measure burnout); and (3) to reduce a data set to a more manageable size while retaining as much of the original information as possible (e.g., we saw in Chapter 7 that multicollinearity can be a problem in multiple regression, and factor analysis can be used to solve this problem by combining variables that are collinear). Through this chapter we'll discover what factors are, how we find them, and what they tell us (if anything) about the relationship between the variables we've measured.

17.3. Factors ②



If we measure several variables, or ask someone several questions about themselves, the correlation between each pair of variables (or questions) can be arranged in what's known as an *R-matrix*. An *R*-matrix is just a correlation matrix: a table of correlation coefficients between variables (in fact, we saw small versions of these matrices in Chapter 6). The diagonal elements of an *R*-matrix are all ones because each variable will correlate perfectly with itself. The off-diagonal elements are the correlation coefficients between pairs of variables, or questions.[1] The existence of clusters of large correlation coefficients between subsets of variables suggests that those variables could be measuring aspects of the same underlying dimension. These underlying dimensions are known as factors (or *latent variables*). By reducing a data set from a group of interrelated variables into a smaller set of factors, factor analysis achieves parsimony by explaining the maximum amount of common variance in a correlation matrix using the smallest number of explanatory constructs.

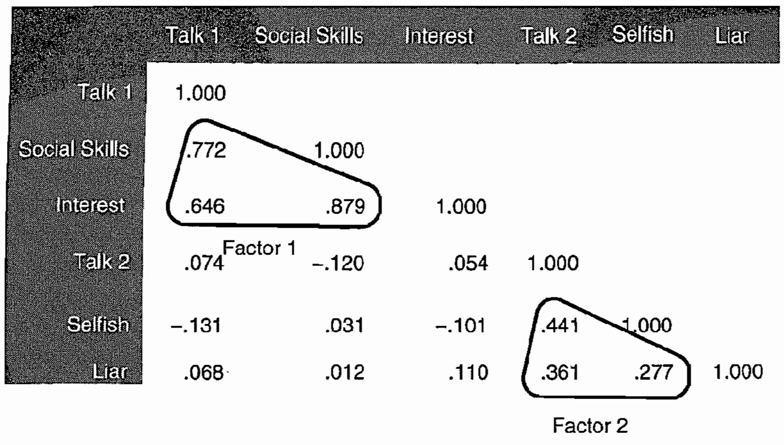
There are numerous examples of the use of factor analysis in the social sciences. The trait theorists in psychology used factor analysis endlessly to assess personality traits. Most readers will be familiar with the extroversion–introversion and neuroticism traits measured by Eysenck (1953). Most other personality questionnaires are based on factor analysis – notably Cattell's (1966a) 16 personality factors questionnaire – and these inventories are frequently used for recruiting purposes in industry (and even by some religious groups). However, although factor analysis is probably most famous for being adopted by psychologists, its use is by no means restricted to measuring dimensions of personality. Economists, for example, might use factor analysis to see whether productivity, profits and workforce can be reduced down to an underlying dimension of company growth.

Let's put some of these ideas into practice by imagining that we wanted to measure different aspects of what might make a person popular. We could administer several measures that we believe tap different aspects of popularity. So, we might measure a person's social skills (Social Skills), their selfishness (Selfish), how interesting others find them (Interest), the proportion of time they spend talking about the other person during a conversation (Talk1), the proportion of time they spend talking about themselves (Talk2), and their propensity to lie to people (the Liar scale). We can then calculate the correlation coefficients for each pair of variables and create an *R*-matrix. Figure 17.2 shows this matrix. Any significant correlation coefficients are shown in bold type. It is clear that there are two clusters of interrelating variables. Therefore, these variables might be measuring some common underlying dimension. The amount that someone talks about the other person during a conversation seems to correlate highly with both the level of social skills and how interesting the other finds that person. Also, social skills correlate well with how interesting others perceive a person to be. These relationships indicate that the better your social skills, the more interesting and talkative you are likely to be. However, there is a second cluster of variables. The amount that people talk about themselves within a conversation correlates with how selfish they are and how much they lie. Being selfish also correlates with the degree to which a person tells lies. In short, selfish people are likely to lie and talk about themselves.

In factor analysis we strive to reduce this *R*-matrix down into its underlying dimensions by looking at which variables seem to cluster together in a meaningful way. This data

[1] This matrix is called an *R*-matrix, or just *R*, because it contains correlation coefficients and *r* usually denotes Pearson's correlation (see Chapter 6) – the *r* turns into a capital letter when it denotes a matrix. Given that this book is about some software called R, this is slightly confusing, so be careful – it should be obvious when we are talking about the program, and when I'm talking about the correlation matrix, and when it's not, I'll tell you.

FIGURE 17.2
An *R*-matrix



	Talk 1	Social Skills	Interest	Talk 2	Selfish	Liar
Talk 1	1.000					
Social Skills	.772	1.000				
Interest	.646	.879	1.000			
Talk 2	.074	−.120	.054	1.000		
Selfish	−.131	.031	−.101	.441	1.000	
Liar	.068	.012	.110	.361	.277	1.000

Factor 1
Factor 2

reduction is achieved by looking for variables that correlate highly with a group of other variables, but do not correlate with variables outside of that group. In this example, there appear to be two clusters that fit the bill. The first factor seems to relate to general sociability, whereas the second factor seems to relate to the way in which a person treats others socially (we might call it 'consideration'). It might, therefore, be assumed that popularity depends not only on your ability to socialize, but also on whether you are genuine towards others.

17.3.1. Graphical representation of factors ②

Factors (not to be confused with independent variables in factorial ANOVA) are statistical entities that can be visualized as classification axes along which measurement variables can be plotted. In plain English, this statement means that if you imagine factors as being the axis of a graph, then we can plot variables along these axes. The coordinates of variables along each axis represent the strength of relationship between that variable and each factor. Figure 17.3 shows such a plot for the popularity data (in which there were only two factors). The first thing to notice is that for both factors, the axis line ranges from −1 to 1, which are the outer limits of a correlation coefficient. Therefore, the position of a given variable depends on its correlation with the two factors. The circles represent the three variables that correlate highly with factor 1 (Sociability: horizontal axis) but have a low correlation with factor 2 (Consideration: vertical axis). Conversely, the triangles represent variables that correlate highly with consideration to others but have a low correlation with sociability. From this plot, we can tell that selfishness, the amount a person talks about themselves and their propensity to lie all contribute to a factor that could be called consideration of others. Conversely, how much a person takes an interest in other people, how interesting they are and their level of social skills contribute to a second factor, sociability. This diagram therefore supports the structure that was apparent in the *R*-matrix. Of course, if a third factor existed within these data it could be represented by a third axis (creating a 3-D graph). It should also be apparent that if more than three factors exist in a data set, then a 2-D drawing cannot represent them all.

If each axis on the graph represents a factor, then the variables that go to make up a factor can be plotted according to the extent to which they relate to a given factor. The coordinates of a variable, therefore, represent its relationship to the factors. In an ideal world a variable should have a large coordinate for one of the axes, and low coordinates for any other factors. This scenario would indicate that this particular variable related to only one

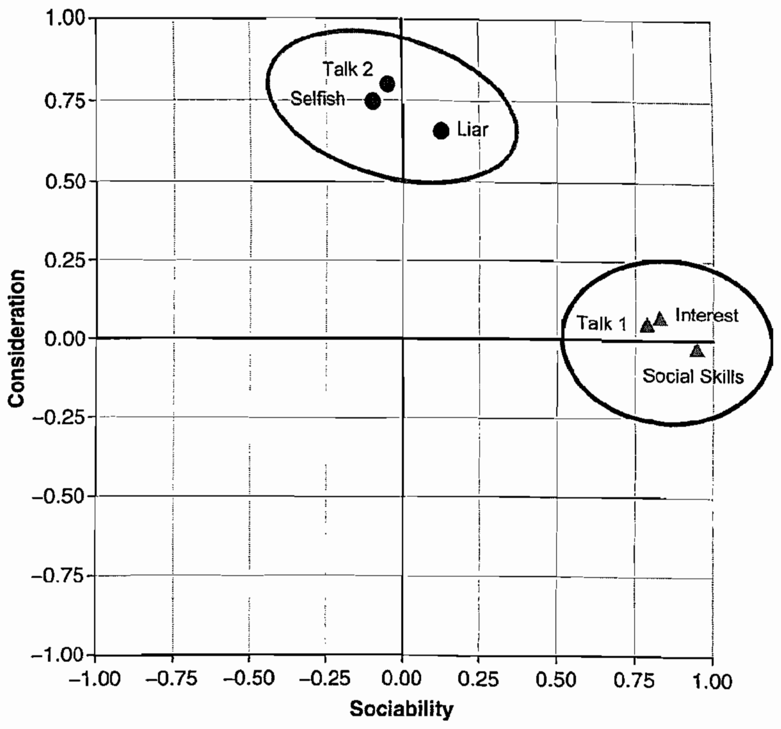


FIGURE 17.3 Example of a factor plot

factor. Variables that have large coordinates on the same axis are assumed to measure different aspects of some common underlying dimension. The coordinate of a variable along a classification axis is known as a **factor loading**. The factor loading can be thought of as the Pearson correlation between a factor and a variable (see Jane Superbrain Box 17.1). From what we know about interpreting correlation coefficients (see section 6.5.4.3) it should be clear that if we square the factor loading we obtain a measure of the substantive importance of a particular variable to a factor.

17.3.2. Mathematical representation of factors ②

The axes drawn in Figure 17.3 are straight lines and so can be described mathematically by the equation of a straight line. Therefore, factors can also be described in terms of this equation.

SELF-TEST

✓ What is the equation of a straight line?

The following equation reminds us of the equation describing a linear model and then applies this to the scenario of describing a factor:

$$Y_i = b_1X_{1i} + b_2X_{2i} + \ldots + b_nX_{ni} + \varepsilon_i$$

$$\text{Factor}_i = b_1\,\text{Variable}_{1i} + b_2\,\text{Variable}_{2i} + \ldots + b_n\,\text{Variable}_{ni} + \varepsilon_i \tag{17.1}$$

You'll notice that there is no intercept in the equation, the reason being that the lines intersect at zero (hence the intercept is also zero). The *b*s in the equation represent the factor loadings.

Sticking with our example of popularity, we found that there were two factors underlying this construct: general sociability and consideration. We can, therefore, construct an equation that describes each factor in terms of the variables that have been measured. The equations are as follows:

$$\begin{aligned} Y_i &= b_1X_{1i} + b_2X_{2i} + \ldots + b_nX_{ni} + \varepsilon_i \\ \text{Sociability}_i &= b_1\text{Talk1}_i + b_2\text{Social Skills}_i + b_3\text{Interest}_i \\ &\quad + b_4\text{Talk2}_i + b_5\text{Selfish}_i + b_6\text{Liar}_i + \varepsilon_i \\ \text{Consideration}_i &= b_1\text{Talk1}_i + b_2\text{Social Skills}_i + b_3\text{Interest}_i \\ &\quad + b_4\text{Talk2}_i + b_5\text{Selfish}_i + b_6\text{Liar}_i + \varepsilon_i \end{aligned} \tag{17.2}$$

Notice that the equations are identical in form: they both include all of the variables that were measured. However, the values of *b* in the two equations will be different (depending on the relative importance of each variable to the particular factor). In fact, we can replace each value of *b* with the coordinate of that variable on the graph in Figure 17.3 (i.e., replace the values of *b* with the factor loading). The resulting equations are as follows:

$$\begin{aligned} Y_i &= b_1X_{1i} + b_2X_{2i} + \ldots + b_nX_{ni} + \varepsilon_i \\ \text{Sociability}_i &= 0.87\text{Talk1}_i + 0.96\text{Social Skills}_i + 0.92\text{Interest}_i \\ &\quad + 0.00\text{Talk2}_i - 0.10\text{Selfish}_i + 0.09\text{Liar}_i + \varepsilon_i \\ \text{Consideration}_i &= 0.01\text{Talk1}_i - 0.03\text{Social Skills}_i + 0.04\text{Interest}_i \\ &\quad + 0.82\text{Talk2}_i + 0.75\text{Selfish}_i + 0.70\text{Liar}_i + \varepsilon_i \end{aligned} \tag{17.3}$$

Observe that, for the sociability factor, the values of *b* are high for Talk1, Social Skills and Interest. For the remaining variables (Talk2, Selfish and Liar) the values of *b* are very low (close to 0). This tells us that three of the variables are very important for that factor (the ones with high values of *b*) and three are very unimportant (the ones with low values of *b*). We saw that this point is true because of the way that three variables clustered highly on the factor plot. The point to take on board here is that the factor plot and these equations represent the same thing: the factor loadings in the plot are simply the *b*-values in these equations (but see Jane Superbrain Box 17.1). For the second factor, inconsideration to others, the opposite pattern can be seen in that Talk2, Selfish and Liar all have high values of *b* whereas the remaining three variables have *b*-values close to 0. In an ideal world, variables would have very high *b*-values for one factor and very low *b*-values for all other factors.

These factor loadings can be placed in a matrix in which the columns represent each factor and the rows represent the loadings of each variable on each factor. For the popularity data this matrix would have two columns (one for each factor) and six rows (one for each variable). This matrix, usually denoted *A*, is given by:

$$A = \begin{pmatrix} 0.87 & 0.01 \\ 0.96 & -0.03 \\ 0.92 & 0.04 \\ 0.00 & 0.82 \\ -0.10 & 0.75 \\ 0.09 & 0.70 \end{pmatrix}$$

To understand what the matrix means, try relating the elements to the loadings in equation (17.3). For example, the top row represents the first variable, Talk1, which had a loading of .87 for the first factor (Sociability) and a loading of .01 for the second factor (Consideration). This matrix is called the **factor matrix** or **component matrix** (if doing **principal components analysis**) – see Jane Superbrain Box 17.1 to find out about the different forms of this matrix.

The major assumption in factor analysis is that these algebraic factors represent real-world dimensions, the nature of which must be *guessed at* by inspecting which variables have high loads on the same factor. So, psychologists might believe that factors represent dimensions of the psyche, education researchers might believe they represent abilities, and sociologists might believe they represent races or social classes. However, it is an extremely contentious point whether this assumption is tenable, and some believe that the dimensions derived from factor analysis are real only in the statistical sense – and are real-world fictions.

JANE SUPERBRAIN 17.1

What's the difference between a pattern matrix and a structure matrix? ③

Throughout my discussion of factor loadings I've been quite vague. Sometimes I've said that these loadings can be thought of as the correlation between a variable and a given factor, then at other times I've described these loadings in terms of regression coefficients (*b*). Now, it should be obvious from what we discovered in Chapters 6 and 7 that correlation coefficients and regression coefficients are quite different things, so what the hell am I going on about: shouldn't I make up my mind what the factor loadings actually are?

Well, in vague terms (the best terms for my brain) both correlation coefficients and regression coefficients represent the relationship between a variable and linear model in a broad sense, so the key take-home message is that factor loadings tell us about the relative contribution that a variable makes to a factor. As long as you understand that much, you have no problems.

However, the factor loadings in a given analysis can be both correlation coefficients and regression coefficients. Soon we'll discover that the interpretation of factor analysis is helped greatly by a technique known as *rotation*. Without going into details, there are two types: orthogonal and oblique rotation (see section 17.3.9). When orthogonal rotation is used, any underlying factors are assumed to be independent, and the factor loading *is* the correlation between the factor and the variable, but is also the regression coefficient. Put another way, the values of the correlation coefficients are the same as the values of the regression coefficients. However, there are situations in which the underlying factors are assumed to be related or correlated to each other. In these situations, oblique rotation is used and the resulting correlations between variables and factors will differ from the corresponding regression coefficients. In this case, there are, in effect, two different sets of factor loadings: the correlation coefficients between each variable and factor (which are put in the factor **structure matrix**) and the regression coefficients for each variable on each factor (which are put in the factor **pattern matrix**). These coefficients can have quite different interpretations (see Graham, Guthrie, & Thompson, 2003).

17.3.3. Factor scores ②

A factor can be described in terms of the variables measured and the relative importance of them for that factor (represented by the value of *b*). Therefore, having discovered which factors exist, and estimated the equation that describes them, it should be possible to also estimate a person's score on a factor, based on their scores for the constituent variables. These scores are known as **factor scores**. As such, if we wanted to derive a score of sociability for a particular person, we could place their scores on the various measures into equation (17.3). This method is known as a *weighted average*. In fact, this method is overly simplistic and rarely used, but it is probably the easiest way to explain the principle. For example, imagine the six scales all range from 1 to 10 and that someone scored the following: Talk1 (4), Social Skills (9), Interest (8), Talk2 (6), Selfish (8), and Liar (6). We could put these values into equation (17.3) to get a score for this person's sociability and their consideration to others:

$$\begin{aligned}\text{Sociability} &= 0.87\text{Talk1} + 0.96\text{SocialSkills} + 0.92\text{Interest}\\ &\quad + 0.00\text{Talk2} - 0.10\text{Selfish} + 0.09\text{Liar}\\ &= (0.87\times 4) + (0.96\times 9) + (0.92\times 8) + (0.00\times 6)\\ &\quad - (0.10\times 8) + (0.09\times 6)\\ &= 19.22\\ \text{Consideration} &= 0.01\text{Talk1} - 0.03\text{SocialSkills} + 0.04\text{Interest}\\ &\quad + 0.82\text{Talk2} + 0.75\text{Selfish} + 0.70\text{Liar}\\ &= (0.01\times 4) - (0.03\times 9) + (0.04\times 8) + (0.82\times 6)\\ &\quad + (0.75\times 8) + (0.70\times 6)\\ &= 15.21\end{aligned} \tag{17.4}$$

The resulting scores of 19.22 and 15.21 reflect the degree to which this person is sociable and their inconsideration to others, respectively. This person scores higher on sociability than inconsideration. However, the scales of measurement used will influence the resulting scores, and if different variables use different measurement scales, then factor scores for different factors cannot be compared. As such, this method of calculating factor scores is poor and more sophisticated methods are usually used.

17.3.3.1. The regression method ④

There are several sophisticated techniques for calculating factor scores that use factor score coefficients as weights in equation (17.1) rather than using the factor loadings. The form of the equation remains the same, but the *b*s in the equation are replaced with these factor score coefficients. Factor score coefficients can be calculated in several ways. The simplest way is the regression method. In this method the factor loadings are adjusted to take account of the initial correlations between variables; in doing so, differences in units of measurement and variable variances are stabilized.

To obtain the matrix of factor score coefficients (B) we multiply the matrix of factor loadings by the inverse (R^{-1}) of the original correlation or R-matrix. You might remember from the previous chapter that matrices cannot be divided (see section 16.4.4.1). Therefore, if we want to divide by a matrix it cannot be done directly and instead we multiply by its inverse. Therefore, by multiplying the matrix of factor loadings by the inverse of the correlation matrix we are, conceptually speaking, dividing the factor loadings by the correlation coefficients. The resulting factor score matrix, therefore, represents the relationship between each variable and each factor, taking into account the original relationships between pairs of variables. As such, this matrix represents a purer measure of the *unique* relationship between variables and factors.

The matrices for the popularity data are shown below. The resulting matrix of factor score coefficients, B, comes from the R (the program) output. The matrices R^{-1} and A can be multiplied by hand to get the matrix B, and those familiar with matrix algebra – or who have consulted Namboodiri, (1984) or Stevens (2002) – might like to verify the result (see Oliver Twisted). To get the same degree of accuracy as **R** you should work to at least five decimal places:

$$B = R^{-1}A$$

$$B = \begin{pmatrix} 4.76 & -7.46 & 3.91 & -2.35 & 2.42 & -0.49 \\ -7.46 & 18.49 & -12.42 & 5.45 & -5.54 & 1.22 \\ 3.91 & -12.42 & 10.07 & -3.65 & 3.79 & -0.96 \\ -2.35 & 5.45 & -3.65 & 2.97 & -2.16 & 0.02 \\ 2.42 & -5.54 & 3.79 & -2.16 & 2.98 & -0.56 \\ -0.49 & 1.22 & -0.96 & 0.02 & -0.56 & 1.27 \end{pmatrix} \begin{pmatrix} 0.87 & 0.01 \\ 0.96 & -0.03 \\ 0.92 & 0.04 \\ 0.00 & 0.82 \\ -0.10 & 0.75 \\ 0.09 & 0.70 \end{pmatrix}$$

$$B = \begin{pmatrix} 0.343 & 0.006 \\ 0.376 & -0.020 \\ 0.362 & 0.020 \\ 0.000 & 0.473 \\ -0.037 & 0.437 \\ 0.039 & 0.405 \end{pmatrix}$$

The pattern of the loadings is the same for the factor score coefficients: that is, the first three variables have high loadings for the first factor and low loadings for the second, whereas the pattern is reversed for the last three variables. The difference is only in the actual value of the weightings, which are smaller because the correlations between variables are now accounted for. These factor score coefficients can be used to replace the b-values in equation (17.2):

$$\begin{aligned} \text{Sociability} &= 0.343\text{Talk1} + 0.376\text{SocialSkills} + 0.362\text{Interest} \\ &\quad + 0.000\text{Talk2} - 0.037\text{Selfish} + 0.039\text{Liar} \\ &= (0.343 \times 4) + (0.376 \times 9) + (0.362 \times 8) + (0.000 \times 6) \\ &\quad - (0.037 \times 8) + (0.039 \times 6) \\ &= 7.59 \\ \text{Consideration} &= 0.006\text{Talk1} - 0.020\text{SocialSkills} + 0.020\text{Interest} \\ &\quad + 0.473\text{Talk2} + 0.437\text{Selfish} + 0.405\text{Liar} \\ &= (0.006 \times 4) - (0.020 \times 9) + (0.020 \times 8) + (0.473 \times 6) \\ &\quad + (0.437 \times 8) + (0.405 \times 6) \\ &= 8.768 \end{aligned} \tag{17.5}$$

Equation (17.5) shows how these coefficient scores are used to produce two factor scores for each person. In this case, the participant had the same scores on each variable as were used in equation (17.4). The resulting scores are much more similar than when the factor loadings were used as weights because the different variances among the six variables have now been controlled for. The fact that the values are very similar reflects the fact that this person not only scores highly on variables relating to sociability, but is also inconsiderate (i.e., they score equally highly on both factors). This technique for producing factor scores ensures that the resulting scores have a mean of 0 and a variance equal to the squared multiple correlation between the estimated factor scores and the true factor values. However, the downside of the regression method is that the scores can correlate not only with factors other than the one on which they are based, but also with other factor scores from a different orthogonal factor.

OLIVER TWISTED

Please Sir, can I have some more … matrix algebra?

'*The Matrix* …', enthuses Oliver, '… that was a good film. I want to dress in black and glide through the air as though time has stood still. Maybe the matrix of factor scores is as cool as the film.' I think you might be disappointed, Oliver, but we'll give it a shot. The matrix calculations of factor scores are detailed in the additional material for this chapter on the companion website. Be afraid, be very afraid …

17.3.3.2. Uses of factor scores ②

There are several uses of factor scores. First, if the purpose of the factor analysis is to reduce a large set of data into a smaller subset of measurement variables, then the factor scores tell us an individual's score on this subset of measures. Therefore, any further analysis can be carried out on the factor scores rather than the original data. For example, we could carry out a *t*-test to see whether females are significantly more sociable than males using the factor scores for *sociability*. A second use is in overcoming collinearity problems in regression. If, following a multiple regression analysis, we have identified sources of multicollinearity then the interpretation of the analysis is questioned (see section 7.7.2.3). In this situation, we can carry out a principal components analysis on the predictor variables to reduce them down to a subset of uncorrelated factors. The variables causing the multicollinearity will combine to form a factor. If we then rerun the regression but using the factor scores as predictor variables then the problem of multicollinearity should vanish (because the variables are now combined into a single factor).

By now, you should have some grasp of the concept of what a factor is, how it is represented graphically, how it is represented algebraically, and how we can calculate composite scores representing an individual's 'performance' on a single factor. I have deliberately restricted the discussion to a conceptual level, without delving into how we actually find these mythical beasts known as factors. This section will look at how we find factors. Specifically, we will examine different types of method, look at the maths behind one method (principal components), investigate the criteria for determining whether factors are important, and discover how to improve the interpretation of a given solution.

17.3.4. Choosing a method ②

The first thing you need to know is that there are several methods for unearthing factors in your data. The method you chose will depend on what you hope to do with the analysis. Tinsley and Tinsley (1987) give an excellent account of the different methods available. There are two things to consider: whether you want to generalize the findings from your sample to a population and whether you are exploring your data or testing a specific hypothesis. This chapter describes techniques for exploring data using factor analysis. Testing hypotheses about the structures of latent variables and their relationships to each other requires considerable complexity and can be done with packages such as *sem* or *Lavaan* in R.[2] Those interested in hypothesis testing techniques (known as **confirmatory**

[2] The *sem* package is the more straightforward, but is slightly less capable of handling unusual situations than *Lavaan* (*sem* was written by John Fox, who also wrote *R Commander*).

factor analysis) are advised to read Pedhazur and Schmelkin (1991, Chapter 23) for an introduction.

Assuming we want to explore our data, we then need to consider whether we want to apply our findings to the sample collected (descriptive method) or to generalize our findings to a population (inferential methods). When factor analysis was originally developed it was assumed that it would be used to explore data to generate future hypotheses. As such, it was assumed that the technique would be applied to the entire population of interest. Therefore, certain techniques assume that the sample used is the population, and so results cannot be extrapolated beyond that particular sample. Principal components analysis is an example of one of these techniques, as is principal factors analysis (*principal axis factoring*). Principal components analysis and principal factors analysis are the preferred methods and usually result in similar solutions (see section 17.3.6). When these methods are used, conclusions are restricted to the sample collected and generalization of the results can be achieved only if analysis using different samples reveals the same factor structure.

Another approach has been to assume that participants are randomly selected and that the variables measured constitute the population of variables in which we're interested. By assuming this, it is possible to develop techniques from which the results can be generalized from the sample participants to a larger population. However, a constraint is that any findings hold true only for the set of variables measured (because we've assumed this set constitutes the entire population of variables). Techniques in this category include the maximum-likelihood method (see Harman, 1976) and Kaiser's alpha factoring. The choice of method depends largely on what generalizations, if any, you want to make from your data.[3]

17.3.5. Communality ②

Before continuing, it is important that you understand some basic things about the variance within an *R*-matrix. It is possible to calculate the variability in scores (the variance) for any given measure (or variable). You should be familiar with the idea of variance by now and comfortable with how it can be calculated (if not, see Chapter 2). The total variance for a particular variable will have two components: some of it will be shared with other variables or measures (**common variance**) and some of it will be specific to that measure (**unique variance**). We tend to use the term *unique variance* to refer to variance that can be reliably attributed to only one measure. However, there is also variance that is specific to one measure but not reliably so; this variance is called error or **random variance**. The proportion of common variance present in a variable is known as the **communality**. As such, a variable that has no specific variance (or random variance) would have a communality of 1; a variable that shares none of its variance with any other variable would have a communality of 0.

In factor analysis we are interested in finding common underlying dimensions within the data and so we are primarily interested only in the common variance. Therefore, when we run a factor analysis it is fundamental that we know how much of the variance present in our data is common variance. This presents us with a logical impasse: to do the factor analysis we need to know the proportion of common variance present in the data, yet the only way to find out the extent of the common variance is by carrying out a factor analysis. There are two ways to approach this problem. The first is to assume that all of the variance is common variance. As such, we assume that the communality of every variable is 1. By making this assumption we merely transpose our original data into constituent linear components (known as principal components analysis). The second approach is to estimate the

[3] It's worth noting at this point that principal component analysis is not in fact the same as factor analysis. This doesn't stop idiots like me from discussing them as though they are, but more on that later.

amount of common variance by estimating communality values for each variable. There are various methods of estimating communalities, but the most widely used (including **alpha factoring**) is to use the squared multiple correlation (SMC) of each variable with all others. So, for the popularity data, imagine you ran a multiple regression using one measure (Selfish) as the outcome and the other five measures as predictors: the resulting multiple R^2 (see section 7.6.2) would be used as an estimate of the communality for the variable Selfish. This second approach is used in factor analysis. These estimates allow the factor analysis to be done. Once the underlying factors have been extracted, new communalities can be calculated that represent the multiple correlation between each variable and the factors extracted. Therefore, the communality is a measure of the proportion of variance explained by the extracted factors.

17.3.6. Factor analysis vs. principal components analysis ②

I have just explained that there are two approaches to locating underlying dimensions of a data set: factor analysis and principal components analysis. These techniques differ in the communality estimates that are used. Simplistically, though, factor analysis derives a mathematical model from which factors are estimated, whereas principal components analysis merely decomposes the original data into a set of linear variates – see Dunteman, (1989) and Widaman (2007) for more detail on the differences between the procedures. As such, only factor analysis can estimate the underlying factors, and it relies on various assumptions for these estimates to be accurate. Principal components analysis is concerned only with establishing which linear components exist within the data and how a particular variable might contribute to that component. In terms of theory, this chapter is dedicated to principal components analysis rather than factor analysis. The reasons are that principal components analysis is a psychometrically sound procedure, is conceptually less complex than factor analysis, and bears numerous similarities to discriminant analysis (described in the previous chapter).

However, we should consider whether the techniques provide different solutions to the same problem. Based on an extensive literature review, Guadagnoli and Velicer (1988) concluded that the solutions generated from principal components analysis differ little from those derived from factor analysis techniques. In reality, there are some circumstances for which this statement is untrue. Stevens (2002) summarizes the evidence and concludes that, with 30 or more variables and communalities greater than .7 for all variables, different solutions are unlikely; however, with fewer than 20 variables and any low communalities (< .4), differences can occur.

The flip-side of this argument is eloquently described by Cliff (1987) who observed that proponents of factor analysis 'insist that components analysis is at best a common factor analysis with some error added and at worst an unrecognizable hodgepodge of things from which nothing can be determined' (p. 349). Indeed, feeling is strong on this issue, with some arguing that when principal components analysis is used it should not be described as a factor analysis and that you should not impute substantive meaning to the resulting components. However, to non-statisticians the difference between a principal component and a factor may be difficult to conceptualize (they are both linear models), and the differences arise largely from the calculation.[4]

[4] For this reason I have used the terms *components* and *factors* interchangeably throughout this chapter. Although this use of terms will reduce some statisticians (and psychologists) to tears, I'm banking on these people not needing to read this book. I acknowledge the methodological differences, but I think it's easier for students if I dwell on the similarities between the techniques and not the differences.

17.3.7. Theory behind principal components analysis ③

Principal components analysis works in a very similar way to MANOVA and discriminant function analysis (see previous chapter). Although it isn't necessary to understand the mathematical principles in any detail, readers of the previous chapter may benefit from some comparisons between the two techniques. For those who haven't read that chapter, I suggest you flick through it before moving ahead!

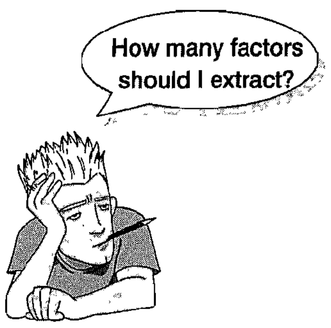
In MANOVA, various sum of squares and cross-product matrices were calculated that contained information about the relationships between dependent variables. I mentioned before that these SSCP matrices could be easily converted to variance–covariance matrices, which represent the same information but in averaged form (i.e., taking account of the number of observations). I also said that by dividing each element by the relevant standard deviation the variance–covariance matrices become standardized. The result is a correlation matrix. In principal components analysis we usually deal with correlation matrices (although it is possible to analyse a variance–covariance matrix too), and the point to note is that this matrix pretty much represents the same information as an SSCP matrix in MANOVA. The difference is just that the correlation matrix is an averaged version of the SSCP that has been standardized.

In MANOVA, we used several SSCP matrices that represented different components of experimental variation (the model variation and the residual variation). In principal components analysis the covariance (or correlation) matrix cannot be broken down in this way (because all data come from the same group of participants). In MANOVA, we ended up looking at the variates or components of the SSCP matrix that represented the ratio of the model variance to the error variance. These variates were linear dimensions that separated the groups tested, and we saw that the dependent variables mapped onto these underlying components. In short, we looked at whether the groups could be separated by some linear combination of the dependent variables. These variates were found by calculating the eigenvectors of the SSCP. The number of variates obtained was the smaller of p (the number of dependent variables) and $k - 1$ (where k is the number of groups). In component analysis we do something similar (I'm simplifying things a little, but it will give you the basic idea). That is, we take a correlation matrix and calculate the variates. There are no groups of observations, and so the number of variates calculated will always equal the number of variables measured (p). The variates are described, as for MANOVA, by the eigenvectors associated with the correlation matrix. The elements of the eigenvectors are the weights of each variable on the variate (see equation (16.5)). These values are the factor loadings described earlier. The largest eigenvalue associated with each of the eigenvectors provides a single indicator of the substantive importance of each variate (or component). The basic idea is that we retain factors with relatively large eigenvalues and ignore those with relatively small eigenvalues.

The eigenvalue for a factor can also be calculated by summing the square of the loadings for that factor. This isn't much use if you're calculating factor analysis, because you need to calculate the eigenvalues to calculate the loadings. But it can be a useful way to help understand the eigenvalues – the higher the loadings on a factor, the more of the variance in the variables that the factor explains.

In summary, component analysis works in a similar way to MANOVA. We begin with a matrix representing the relationships between variables. The linear components (also called variates, or factors) of that matrix are then calculated by determining the eigenvalues of the matrix. These eigenvalues are used to calculate eigenvectors, the elements of which provide the loading of a particular variable on a particular factor (i.e., they are the b-values in equation (17.1)). The eigenvalue is also a measure of the substantive importance of the eigenvector with which it is associated.

17.3.8. Factor extraction: eigenvalues and the scree plot ②



Not all factors are retained in an analysis, and there is debate over the criterion used to decide whether a factor is statistically important. I mentioned above that the eigenvalues associated with a variate indicate the substantive importance of that factor. Therefore, it seems logical that we should retain only factors with large eigenvalues. Retaining factors is known as factor **extraction**. How do we decide whether or not an eigenvalue is large enough to represent a meaningful factor? Well, one technique advocated by Cattell (1966b) is to plot a graph of each eigenvalue (Y-axis) against the factor with which it is associated (X-axis). This graph is known as a **scree plot** (because it looks like a rock face with a pile of debris, or scree, at the bottom). I mentioned earlier that it is possible to obtain as many factors as there are variables and that each has an associated eigenvalue. By graphing the eigenvalues, the relative importance of each factor becomes apparent. Typically there will be a few factors with quite high eigenvalues, and many factors with relatively low eigenvalues, and so this graph has a very characteristic shape: there is a sharp descent in the curve followed by a tailing off (see Figure 17.4). Cattell (1966b) argued that the cut-off point for selecting factors should be at the point of inflexion of this curve. The point of inflexion is where the slope of the line changes dramatically: so, in Figure 17.4, imagine drawing a straight line that summarizes the vertical part of the plot and another that summarizes the horizontal part (the blue dashed lines); then the point of inflexion is the data point at which these two lines meet. In both examples in Figure 17.4 the point of inflexion occurs at the third data point (factor); therefore, we would extract two factors. Thus, you retain (or extract) only factors to the left of the point of inflexion (and do not include the factor at the point of inflexion itself).[5] With a sample of more than 200 participants, the scree plot provides a fairly reliable criterion for factor selection (Stevens, 2002).

Although scree plots are very useful, factor selection should not be based on this criterion alone. Kaiser (1960) recommended retaining all factors with eigenvalues greater than 1. This criterion is based on the idea that the eigenvalues represent the amount of variation explained by a factor and that an eigenvalue of 1 represents a substantial amount of variation. Jolliffe (1972, 1986) reports that **Kaiser's criterion** is too strict and suggests the third option of retaining all factors with eigenvalues greater than .7. The difference between how many factors are retained using Kaiser's methods compared to Jolliffe's can be dramatic.

You might well wonder how the methods compare. Generally speaking, Kaiser's criterion overestimates the number of factors to retain (see Jane Superbrain Box 17.2) but there is some evidence that it is accurate when the number of variables is less than 30 and the resulting communalities (after extraction) are all greater than .7. Kaiser's criterion can also be accurate when the sample size exceeds 250 and the average communality is greater than or equal to .6. In any other circumstances you are best advised to use a scree plot provided the sample size is greater than 200 (see Stevens, 2002, for more detail).

[5] Actually, in his original paper, Cattell advised including the factor at the point of inflexion as well because it is 'desirable to include at least one common error factor as a "garbage can"'. The idea is that the point of inflexion represents an error factor. However, in practice this garbage can factor is rarely retained; also Thurstone argued that it is better to retain too few than too many factors, so most people do *not* retain the factor at the point of inflexion.

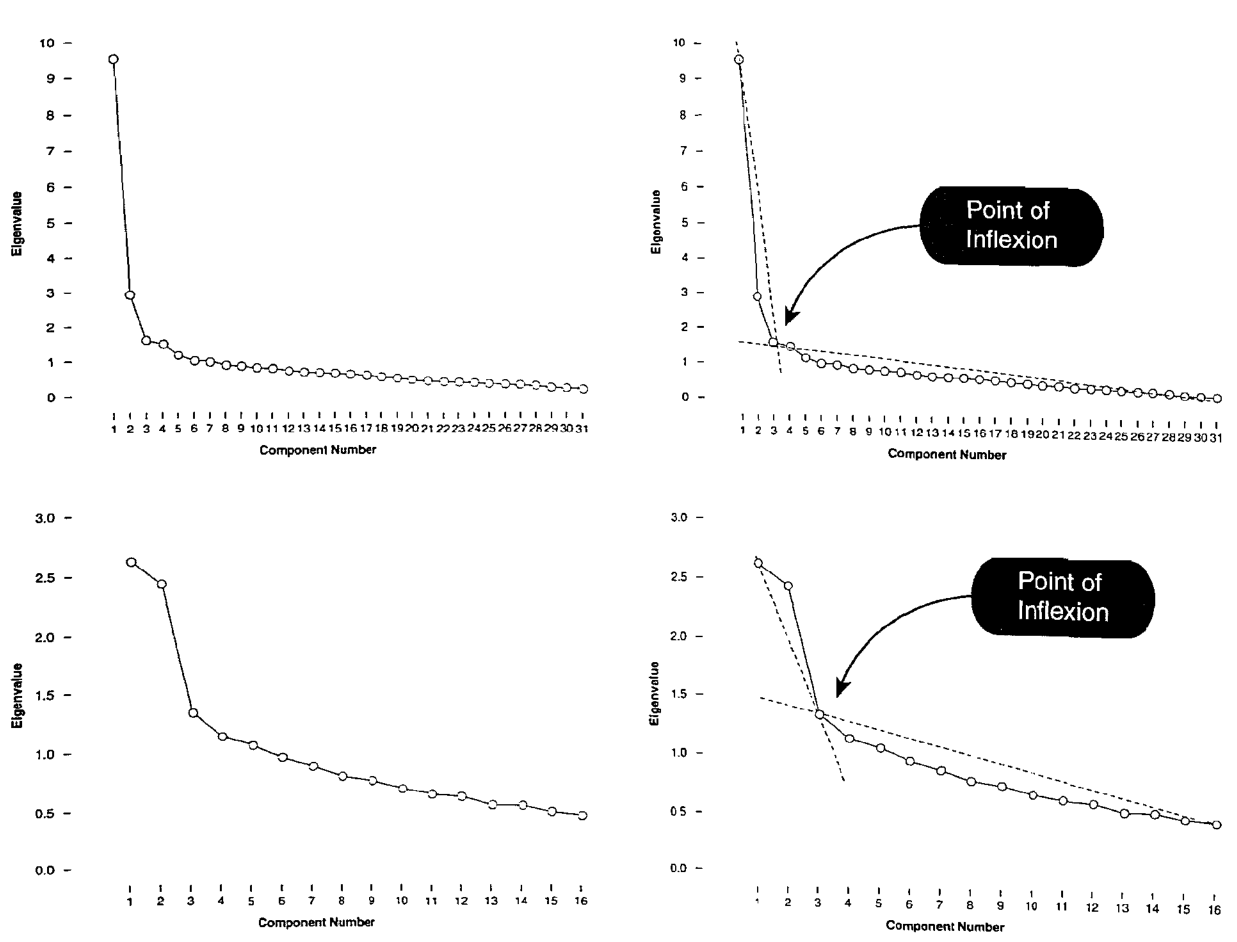


FIGURE 17.4
Examples of scree plots for data that probably have two underlying factors

However, as is often the case in statistics, the three criteria often provide different solutions. In these situations the communalities of the factors need to be considered. In principal components analysis we begin with communalities of 1 with all factors retained (because we assume that all variance is common variance). At this stage all we have done is to find the linear variates that exist in the data – so we have just transformed the data without discarding any information. However, to discover what common variance *really* exists between variables we must decide which factors are meaningful and discard any that are too trivial to consider. Therefore, we discard some information. The factors we retain will not explain all of the variance in the data (because we have discarded some information) and so the communalities after extraction will always be less than 1. The factors retained do not map perfectly onto the original variables – they merely reflect the common variance present in the data. If the communalities represent a loss of information then they are important statistics. The closer the communalities are to 1, the better our factors are at explaining the original data. It is logical that the greater the number of factors retained, the greater the communalities will be (because less information is discarded); therefore, the communalities are good indices of whether too few factors have been retained. In fact, with generalized least-squares factor analysis and maximum-likelihood factor analysis you can get a statistical measure of the goodness of fit of the factor solution (see the next chapter for more on goodness-of-fit tests). This basically measures the proportion of variance that the factor solution explains (so can be thought of as comparing communalities before and after extraction).

As a final word of advice, your decision on how many factors to extract will depend also on why you're doing the analysis; for example, if you're trying to overcome multicollinearity problems in regression, then it might be better to extract too many factors than too few.

JANE SUPERBRAIN 17.2

How many factors do I retain? ③

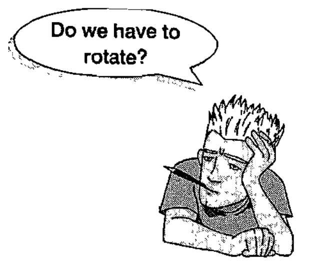
The discussion of factor extraction in the text is somewhat simplified. In fact, there are fundamental problems with Kaiser's criterion (Nunnally & Bernstein, 1994; Preacher & MacCallum, 2003). For one thing an eigenvalue of 1 means different things in different analyses: with 100 variables it means that a factor explains 1% of the variance, but with 10 variables it means that a factor explains 10% of the variance. Clearly, these two situations are very different and a single rule that covers both is inappropriate. An eigenvalue of 1 also means only that the factor explains as much variance as a variable, which rather defeats the original intention of the analysis to reduce variables down to 'more substantive' underlying factors (Nunnally & Bernstein, 1994). Consequently, Kaiser's criterion often overestimates the number of factors. On this basis Jolliffe's criterion is even worse (a factor explains less variance than a variable!).

There are other ways to determine how many factors to retain, but they are more complex (which is why I'm discussing them outside of the main text). The best is probably parallel analysis (Horn, 1965). Essentially each eigenvalue (which represents the size of the factor) is compared against an eigenvalue for the corresponding factor in many randomly generated data sets that have the same characteristics as the data being analysed. In doing so, each eigenvalue is being compared to an eigenvalue from a data set that has no underlying factors. This is a bit like asking whether our observed factor is bigger than a non-existing factor. Factors that are bigger than their 'random' counterparts are retained. Of parallel analysis, the scree plot and Kaiser's criterion, Kaiser's criterion is, in general, worst and parallel analysis best (Zwick & Velicer, 1986).

17.3.9. Improving interpretation: factor rotation ③

Once factors have been extracted, it is possible to calculate to what degree variables load on these factors (i.e., calculate the loading of the variable on each factor). Generally, you will find that most variables have high loadings on the most important factor and small loadings on all other factors. This characteristic makes interpretation difficult, and so a technique called factor rotation is used to discriminate between factors. If a factor is a classification axis along which variables can be plotted, then factor **rotation** effectively rotates these factor axes such that variables are loaded maximally on only one factor. Figure 17.5 demonstrates how this process works using an example in which there are only two factors.

Imagine that a sociologist was interested in classifying university lecturers as a demographic group. She discovered that two underlying dimensions best describe this group: alcoholism and achievement (go to any academic conference and you'll see that academics drink heavily). The first factor, alcoholism, has a cluster of variables associated with it (dark blue circles), and these could be measures such as the number of units drunk in a week, dependency and obsessive personality. The second factor, achievement, also has a cluster of variables associated with it (light blue circles), and these could be measures relating to salary, job status and number of research publications. Initially, the full lines represent the factors, and by looking at the coordinates it should be clear that the light blue circles have high loadings for factor 2 (they are a long way up this axis) and medium loadings for factor 1 (they are not very far up this axis). Conversely, the dark blue circles have high loadings for factor 1 and medium loadings for factor 2. By rotating the axes (dashed lines), we ensure that both clusters of variables are intersected by the factor to which they relate most. So, after rotation, the loadings of the variables are maximized on one factor (the factor that intersects the cluster) and minimized on the remaining factor(s). If an axis passes through a cluster of variables, then these variables will have a loading of approximately zero on the opposite axis. If this idea is confusing, then look at Figure 17.5 and think about the values of the coordinates before and after rotation (this is best achieved by turning the book when you look at the rotated axes).



There are two types of rotation that can be done. The first is **orthogonal rotation**, and the left-hand side of Figure 17.5 represents this method. In Chapter 10 we saw that the term *orthogonal* means unrelated, and in this context it means that we rotate factors while keeping them independent, or unrelated. Before rotation, all factors are independent (i.e., they do not correlate at all) and orthogonal rotation ensures that the factors remain uncorrelated. That is why in Figure 17.5 the axes are turned while remaining perpendicular.[6] The other form of rotation is **oblique rotation**. The difference with oblique rotation is that the factors are allowed to correlate (hence, the axes of the right-hand diagram of Figure 17.5 do not remain perpendicular).

The choice of rotation depends on whether there is a good theoretical reason to suppose that the factors should be related or independent (but see my later comments on this), and also how the variables cluster on the factors before rotation. On the first point, we might not expect alcoholism to be completely independent of achievement (after all, high achievement leads to high stress, which can lead to the drinks cabinet!). Therefore, on theoretical grounds, we might choose oblique rotation. On the second point, Figure 17.5 demonstrates how the positioning of clusters is important in determining how successful the rotation will be (note the position of the light blue circles). Specifically, if an orthogonal rotation was carried out on the right-hand diagram it would be considerably less successful in maximizing loadings than the oblique rotation that is displayed. One approach is to run the analysis using both types of rotation. Pedhazur and Schmelkin (1991) suggest that if the oblique rotation demonstrates a negligible correlation between the extracted factors then it is reasonable to use the orthogonally rotated solution. If the oblique rotation reveals a correlated factor structure, then the orthogonally rotated solution should be discarded. In

[6] This term means that the axes are at right angles to one another.

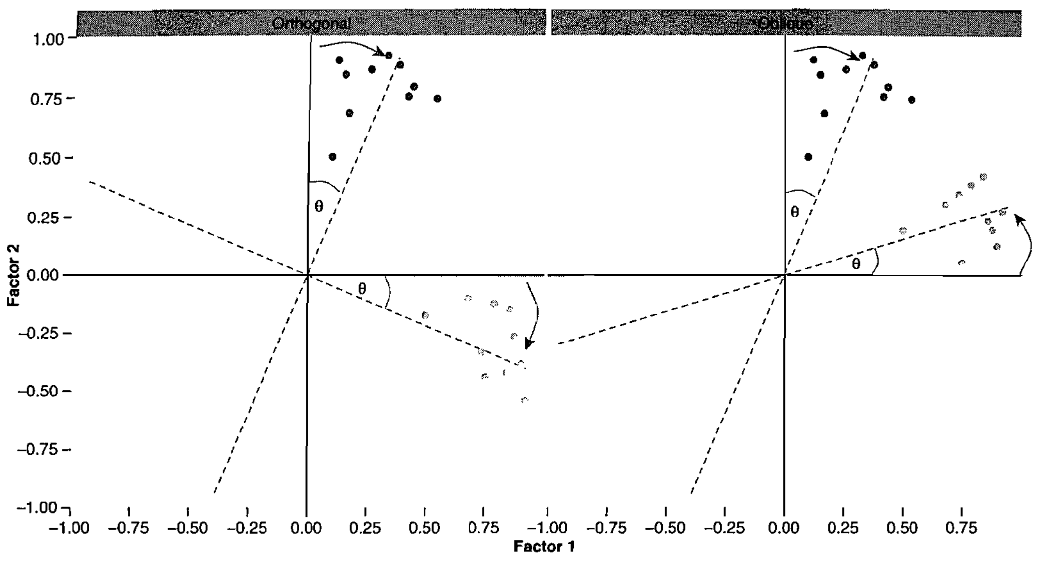


FIGURE 17.5 Schematic representations of factor rotation. The left graph displays orthogonal rotation whereas the right graph displays oblique rotation (see text for more details). θ is the angle through which the axes are rotated

any case, an oblique rotation should be used only if there are good reasons to suppose that the underlying factors *could* be related in theoretical terms.

The mathematics behind factor rotation is complex (especially oblique rotation). However, in oblique rotation, because each factor can be rotated by different amounts, a **factor transformation matrix**, Λ is needed. The factor transformation matrix is a square matrix and its size depends on how many factors were extracted from the data. If two factors are extracted then it will be a 2×2 matrix, but if four factors are extracted then it becomes a 4×4 matrix. The values in the factor transformation matrix consist of sines and cosines of the angle of axis rotation (θ). This matrix is multiplied by the matrix of unrotated factor loadings, A, to obtain a matrix of rotated factor loadings.

For the case of two factors the factor transformation matrix would be:

$$\Lambda = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

Therefore, you should think of this matrix as representing the angle through which the axes have been rotated, or the degree to which factors have been rotated. The angle of rotation necessary to optimize the factor solution is found in an iterative way (see R's Souls' Tip 8.1) and different methods can be used.

17.3.9.1. Choosing a method of factor rotation ③

The **R** function that we will use has four methods of orthogonal rotation (**varimax**, **quartimax**, BentlerT and geominT) and five methods of oblique rotation (oblimin, **promax**, simplimax, BentlerQ and geominQ). These methods differ in how they rotate the factors and, therefore, the resulting output depends on which method you select.

The most important orthogonal rotations are quartimax and varimax. Quartimax rotation attempts to maximize the spread of factor loadings for a variable across all factors.

Therefore, interpreting variables becomes easier. However, this often results in lots of variables loading highly on a single factor. Varimax is the opposite in that it attempts to maximize the dispersion of loadings within factors. Therefore, it tries to load a smaller number of variables highly on each factor, resulting in more interpretable clusters of factors. For a first analysis, you should probably select varimax because it is a good general approach that simplifies the interpretation of factors.

The two important oblique rotations are promax and oblimin. Promax is a faster procedure designed for very large data sets. (If you are interested in adjustments that can be made to these rotations, other rotations, and even hand rotations, you can consult the *GPARotate()* function, found in the *psych* package.)

In theory, the exact choice of rotation will depend largely on whether or not you think that the underlying factors should be related. If you expect the factors to be independent then you should choose one of the orthogonal rotations (I recommend varimax). If, however, there are theoretical grounds for supposing that your factors might correlate, then **direct oblimin** should be selected. In practice, there are strong grounds to believe that orthogonal rotations are a complete nonsense for naturalistic data, and certainly for any data involving humans (can you think of any psychological construct that is not in any way correlated with some other psychological construct?). As such, some argue that orthogonal rotations should never be used.

17.3.9.2. Substantive importance of factor loadings ②

Once a factor structure has been found, it is important to decide which variables make up which factors. Earlier I said that the factor loadings were a gauge of the substantive importance of a given variable to a given factor. Therefore, it makes sense that we use these values to place variables with factors. It is possible to assess the statistical significance of a factor loading (after all, it is simply a correlation coefficient or regression coefficient); however, there are various reasons why this option is not as easy as it seems (see Stevens, 2002, p. 393). Typically, researchers take a loading of an absolute value of more than 0.3 to be important. However, the significance of a factor loading will depend on the sample size. Stevens (2002) produced a table of critical values against which loadings can be compared. To summarize, he recommends that for a sample size of 50 a loading of 0.722 can be considered significant, for 100 the loading should be greater than 0.512, for 200 it should be greater than 0.364, for 300 it should be greater than 0.298, for 600 it should be greater than 0.21, and for 1000 it should be greater than 0.162. These values are based on an alpha level of .01 (two-tailed), which allows for the fact that several loadings will need to be tested (see Stevens, 2002, for further detail). Therefore, in very large samples, small loadings can be considered statistically meaningful. (R can provide significance tests of factor loadings, but these get rather complex and are rarely used. By applying Stevens's guidelines you should gain some insight into the structure of variables and factors.)

The significance of a loading gives little indication of the substantive importance of a variable to a factor. This value can be found by squaring the factor loading to give an estimate of the amount of variance in a factor accounted for by a variable (like R^2). In this respect Stevens (2002) recommends interpreting only factor loadings with an absolute value greater than 0.4 (which explain around 16% of the variance in the variable).

17.4. Research example ②

One of the uses of factor analysis is to develop questionnaires: after all, if you want to measure an ability or trait, you need to ensure that the questions asked relate to the construct

FIGURE 17.6
The **R** anxiety questionnaire (RAQ)

SD = Strongly Disagree, D = Disagree, N = Neither, A = Agree, SA = Strongly Agree

		SD	D	N	A	SA
1	Statistics make me cry	O	O	O	O	O
2	My friends will think I'm stupid for not being able to cope with R	O	O	O	O	O
3	Standard deviations excite me	O	O	O	O	O
4	I dream that Pearson is attacking me with correlation coefficients	O	O	O	O	O
5	I don't understand statistics	O	O	O	O	O
6	I have little experience of computers	O	O	O	O	O
7	All computers hate me	O	O	O	O	O
8	I have never been good at mathematics	O	O	O	O	O
9	My friends are better at statistics than me	O	O	O	O	O
10	Computers are useful only for playing games	O	O	O	O	O
11	I did badly at mathematics at school	O	O	O	O	O
12	People try to tell you that R makes statistics easier to understand but it doesn't	O	O	O	O	O
13	I worry that I will cause irreparable damage because of my incompetence with computers	O	O	O	O	O
14	Computers have minds of their own and deliberately go wrong whenever I use them	O	O	O	O	O
15	Computers are out to get me	O	O	O	O	O
16	I weep openly at the mention of central tendency	O	O	O	O	O
17	I slip into a coma whenever I see an equation	O	O	O	O	O
18	R always crashes when I try to use it	O	O	O	O	O
19	Everybody looks at me when I use R	O	O	O	O	O
20	I can't sleep for thoughts of eigenvectors	O	O	O	O	O
21	I wake up under my duvet thinking that I am trapped under a normal distribution	O	O	O	O	O
22	My friends are better at R than I am	O	O	O	O	O
23	If I am good at statistics people will think I am a nerd	O	O	O	O	O

that you intend to measure. I have noticed that a lot of students become very stressed about **R**. Therefore I wanted to design a questionnaire to measure a trait that I termed '**R** anxiety'. I decided to devise a questionnaire to measure various aspects of students' anxiety towards learning **R**. I generated questions based on interviews with anxious and non-anxious students and came up with 23 possible questions to include. Each question was a statement followed by a five-point Likert scale ranging from 'strongly disagree' through 'neither agree nor disagree' to 'strongly agree'. The questionnaire is printed in Figure 17.6.

The questionnaire was designed to predict how anxious a given individual would be about learning how to use **R**. What's more, I wanted to know whether anxiety about **R** could be broken down into specific forms of anxiety. In other words, what latent variables contribute to anxiety about **R**? With a little help from a few lecturer friends I collected 2571 completed questionnaires (at this point it should become apparent that this example is fictitious). The data are stored in the file **RAQ.dat**. Load this file into **R** and have a look at the data. We know that in **R**, cases (or people's data) are typically stored in rows and variables are stored in columns and so this layout is consistent with past chapters. The second thing to notice is that there are 23 variables labelled **Q01** to **Q23**.

OLIVER TWISTED

Please Sir, can I have some more … questionnaires?

'I'm going to design a questionnaire to measure one's propensity to pick a pocket or two', says Oliver, 'but how would I go about doing it?' You'd read the useful information about the dos and don'ts of questionnaire design in the additional material for this chapter on the companion website, that's how. Rate how useful it is on a Likert scale from 1 = not useful at all, to 5 = very useful.

17.4.1. Sample size ②

Correlation coefficients fluctuate from sample to sample, much more so in small samples than in large. Therefore, the reliability of factor analysis is also dependent on sample size. Much has been written about the necessary sample size for factor analysis, resulting in many 'rules of thumb'. The common rule is to suggest that a researcher has at least 10–15 participants per variable. Although I've heard this rule bandied about on numerous occasions, its empirical basis is unclear (although Nunnally, 1978, did recommend having 10 times as many participants as variables). Kass and Tinsley (1979) recommended having between 5 and 10 participants per variable up to a total of 300 (beyond which test parameters tend to be stable regardless of the participant to variable ratio). Indeed, Tabachnick and Fidell (2007) agree that 'it is comforting to have at least 300 cases for factor analysis' (p. 613), and Comrey and Lee (1992) class 300 as a good sample size, 100 as poor and 1000 as excellent.

Fortunately, recent years have seen empirical research done in the form of experiments using simulated data (so-called Monte Carlo studies). Arrindell and van der Ende (1985) used real-life data to investigate the effect of different participant to variable ratios. They concluded that changes in this ratio made little difference to the stability of factor solutions. Guadagnoli and Velicer (1988) found that the most important factors in determining reliable factor solutions were the absolute sample size and the absolute magnitude of factor loadings. In short, they argue that if a factor has four or more loadings greater than .6 then it is reliable regardless of sample size. Furthermore, factors with 10 or more loadings greater than .40 are reliable if the sample size is greater than 150. Finally, factors with a few low loadings should not be interpreted unless the sample size is 300 or more. MacCallum, Widaman, Zhang, and Hong (1999) have shown that the minimum sample size or sample to variable ratio depends on other aspects of the design of the study. In short, their study indicated that as communalities become lower the importance of sample size increases. With all communalities above .6, relatively small samples (less than 100) may be perfectly adequate. With communalities in the .5 range, samples between 100 and 200 can be good enough provided there are relatively few factors each with only a small number of indicator variables. In the worst scenario of low communalities (well below .5) and a larger number of underlying factors they recommend samples above 500.

What's clear from this work is that a sample of 300 or more will probably provide a stable factor solution, but that a wise researcher will measure enough variables to adequately measure all of the factors that theoretically they would expect to find.

Another alternative is to use the **Kaiser–Meyer–Olkin (KMO) measure of sampling adequacy** (Kaiser, 1970). The KMO can be calculated for individual and multiple variables and represents the ratio of the squared correlation between variables to the squared partial correlation between variables. The KMO statistic varies between 0 and 1. A value of 0 indicates that the sum of partial correlations is large relative to the sum of correlations, indicating diffusion in the pattern of correlations (hence, factor analysis is likely to be inappropriate). A value close to 1 indicates that patterns of correlations are relatively compact and so factor analysis should yield distinct and reliable factors. Kaiser (1974) recommends accepting values greater than .5 as barely acceptable (values below this should lead you to either collect more data or rethink which variables to include). Furthermore, values between .5 and .7 are mediocre, values between .7 and .8 are good, values between .8 and .9 are great and values above .9 are superb (Hutcheson & Sofroniou, 1999).

17.4.2. Correlations between variables ③

When I was an undergraduate, my statistics lecturer always used to say 'if you put garbage in, you get garbage out'. This saying applies particularly to factor analysis, because **R** will usually find a factor solution for a set of variables. However, the solution is unlikely to have any real meaning if the variables analysed are not sensible. The first thing to do when conducting a factor analysis or principal components analysis is to look at the correlations of the variables. There are essentially two potential problems: (1) correlations that are not high enough; and (2) correlations that are too high. The correlations between variables can be checked using the *cor()* function (see Chapter 6) to create a correlation matrix of all variables. In both cases the remedy is to remove variables from the analysis. We will look at each problem in turn.

If our test questions measure the same underlying dimension (or dimensions) then we would expect them to correlate with each other (because they are measuring the same thing). Even if questions measure different aspects of the same things (e.g., we could measure overall anxiety in terms of sub-components such as worry, intrusive thoughts and physiological arousal), there should still be high correlations between the variables relating to these subtraits. We can test for this problem first by visually scanning the correlation matrix and looking for correlations below about .3: if any variables have lots of correlations below this value then consider excluding them. It should be immediately clear that this approach is very subjective: I've used fuzzy terms such as 'about .3' and 'lots of', but I have to because every data set is different. Analysing data really is a skill, not a matter of following a recipe book.

If you want an objective test of whether correlations (overall) are too small then you can test for a very extreme scenario. If the variables in our correlation matrix did not correlate at all, then our correlation matrix would be an identity matrix (i.e., the off-diagonal components are zero – see section 16.4.2). **Bartlett's test** examines whether the population correlation matrix resembles an identity matrix. If the population correlation matrix resembles an identity matrix then it means that every variable correlates very badly with all other variables (i.e., all correlation coefficients are close to zero). If it *were* an identity matrix then it would mean that all variables are perfectly independent of one another (all correlation coefficients are zero). Given that we are looking for clusters of variables that measure similar things, it should be obvious why this scenario is problematic: if no variables correlate then there are no clusters to find. Bartlett's test tells us whether our correlation matrix is significantly different from an identity matrix. Therefore, if it is significant then it means that the correlations between variables are (overall) significantly different from zero. So, if Bartlett's test is significant then it is good news. However, as with any significance test, it depends on sample sizes and in factor analysis we typically use very large samples. Therefore, although a non-significant Bartlett's test is certainly cause for concern, a significant test does not necessarily mean that correlations are big enough to make the analysis meaningful. If you do identify any variables that seem to have very low correlations with lots of other variables, then exclude them from the factor analysis.

The opposite problem is when variables correlate too highly. Although mild multicollinearity is not a problem for factor analysis it is important to avoid extreme multicollinearity (i.e., variables that are very highly correlated) and **singularity** (variables that are perfectly correlated). As with regression, multicollinearity causes problems in factor analysis because it becomes impossible to determine the unique contribution to a factor of the variables that are highly correlated (as was the case for multiple regression). Multicollinearity does not cause a problem for principal components analysis. Therefore, as well as scanning the correlation matrix for low correlations, we could also look out for very high correlations ($r > .8$). The problem with a heuristic such as this is that the effect of two variables correlating with $r = .9$ might be less than the effect of, say, three variables that all correlate at $r = .6$. In other words, eliminating such highly correlating variables might not be getting at the cause of the multicollinearity (Rockwell, 1975).

Multicollinearity can be detected by looking at the determinant of the R-matrix, denoted $|R|$ (see Jane Superbrain Box 17.3). One simple heuristic is that the determinant of the R-matrix should be greater than 0.00001.

If you have reason to believe that the correlation matrix has multicollinearity then you could look through the correlation matrix for variables that correlate very highly ($R > .8$) and consider eliminating one of the variables (or more, depending on the extent of the

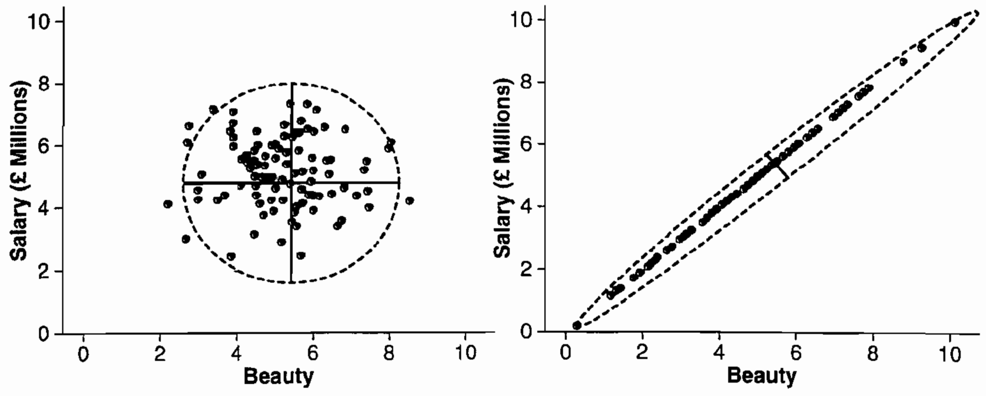
JANE SUPERBRAIN 17.3

What is the determinant? ③

The determinant of a matrix is an important diagnostic tool in factor analysis, but the question of what it is is not easy to answer because it has a mathematical definition and I'm not a mathematician. Rather than pretending that I understand the maths, all I'll say is that a good explanation of how the determinant is derived can be found at http://mathworld.wolfram.com. However, we can bypass the maths and think about the determinant conceptually.

The way that I think of the determinant is as describing the 'area' of the data. In Jane Superbrain Box 16.2 we saw the two diagrams below.

At the time I used these to describe eigenvectors and eigenvalues (which describe the shape of the data). The determinant is related to eigenvalues and eigenvectors, but instead of describing the height and width of the data it describes the overall area. So, in the left diagram below, the determinant of those data would represent the area inside the dashed elipse. These variables have a low correlation so the determinant (area) is big; the biggest value it can be is 1. In the right diagram, the variables are perfectly correlated or singular, and the elipse (dashed line) has been squashed down to basically a straight line. In other words, the opposite sides of the ellipse have actually met each other and there is no distance between them at all. Put another way, the area, or determinant, is zero. Therefore, the determinant tells us whether the correlation matrix is singular (determinant is 0), or if all variables are completely unrelated (determinant is 1), or somewhere in between.



problem) before proceeding. You may have to try some trial and error to work out which variables are creating the problem (it's not always the two with the highest correlation, it could be a larger number of variables with correlations that are not obviously too large).

17.4.3. The distribution of data ②

As well as looking for interrelations, you should ensure that variables have roughly normal distributions and are measured at an interval level (which Likert scales are, perhaps wrongly, assumed to be). The assumption of normality is most important if you wish to generalize the results of your analysis beyond the sample collected. You can do factor analysis on non-continuous data; for example, if you had dichotomous variables you should construct the correlation matrix from polychoric correlation coefficients (these can be calculated using the *polychor()* function, found in the polycor package, which we used in Chapter 6).[7]

17.5. Running the analysis with R Commander ①

If you look through the menus, you'll find 'factor analysis'. The factor analysis that's available in R Commander is a little limited: it does only one kind of extraction (maximum likelihood) and, although this is a good method when it works, if often doesn't work. Understanding why it didn't work and what to do about it is difficult (and the solution is often to just use a different sort of extraction). For this reason, we don't recommend factor analysis with R Commander.

17.6. Running the analysis with R ②

17.6.1. Packages used in this chapter ①

There are several packages we will use in this chapter. You will need the packages *corpcor*, *GPArotation* (for rotating) and *psych* (for the factor analysis). If you don't have these packages installed you'll need to install them and load them.

```
install.packages("corpcor");  install.packages("GPArotation");  install.
packages("psych")
```

Then you need to load the packages by executing these commands:

```
library(corpcor); library(GPArotation); library(psych)
```

17.6.2. Initial preparation and analysis ②

To run a factor analysis or a principal components analysis you can either use the raw data, or you can calculate a correlation matrix, and use that. If you have a *massive* number of

[7] Note that there is an h in the polychor function, that's because we're calculating *polychoric* correlations, using a package that calculates *poly*choric and *poly*serial *cor*relations. (Also note that it's written by John Fox, author of several other packages we use in this book.)

cases (and by massive, I mean at least 100,000, and probably closer to 1,000,000) you're better off calculating a correlation matrix first, and then factor-analysing that. If you don't have a massive number of cases, it doesn't matter which you do. It's also worth noting at this stage that sometimes the analysis doesn't work, usually because the correlation matrix that you're trying to analyse is weird (R's Souls' Tip 17.1).

First, we'll load the data into a dataframe called *raqData*. Set your working directory to the location of the file (see section 3.4.4) and execute:

```
raqData<-read.delim("raq.dat", header = TRUE)
```

We want to include all of the variables in our data set in our factor analysis. We can calculate the correlation matrix, using the *cor()* function (see Chapter 6):

```
raqMatrix<-cor(raqData)
```

R's Souls' Tip 17.1 **Warning messages about non-positive definite matrix ④**

On rare occasions, you might have a non-positive definite matrix. When you have this, **R** will give unhelpful warnings, such as:

```
Warning messages:
1: In log(det(m.inv.r)) : NaNs produced
2: In log(det(r)) : NaNs produced
```

What **R** is trying to tell you, in it's own friendly way, is that the determinant of the R (correlation) matrix is negative, and hence it cannot find the log of the determinant ('NaN' is **R**'s way of saying "not a number"). This problem is usually described as a non-positive definite matrix.

What is a non-positive definite matrix? As we have seen, factor analysis works by looking at your correlation matrix. This matrix has to be 'positive definite' for the analysis to work. What does that mean in plain English? It means lots of horrible things mathematically (e.g., the eigenvalues and determinant of the matrix have to be positive) and about the best explanation I've seen is at http://www2.gsu.edu/~mkteer/npdmatri.html. In more basic terms, factors are like lines floating in space, and eigenvalues measure the length of those lines. If your eigenvalue is negative then it means that the length of your line/factor is negative too. It's a bit like me asking you how tall you are, and you responding 'I'm minus 175 cm tall'. That would be nonsense. By analogy, if a factor has negative length, then that too is nonsense. When **R** decomposes the correlation matrix to look for factors, if it comes across a negative eigenvalue it starts thinking 'oh dear, I've entered some weird parallel universe where the usual rules of maths no longer apply and things can have negative lengths, and this probably means that time runs backwards, my mum is my dad, my sister is a dog, my head is a fish, and my toe is a frog called Gerald'. It still has a go at producing results, but those results probably won't make much sense. (We'd like it if it said 'these results are probably nonsense', rather than being a bit subtle about it, so you have to be *really* careful.)

Things like the KMO test and the determinant rely on a positive definite matrix; if you don't have one they can't be computed.

Why have I got a non-positive definite matrix? The most likely answer is that you have too many variables and too few cases of data, which makes the correlation matrix a bit unstable. It could also be that you have too many highly correlated items in your matrix (singularity, for example, tends to mess things up). In any case it means that your data are bad, naughty data, and not to be trusted; if you let them loose then you have only yourself to blame for the consequences.

What can I do? Other than cry, there's not that much you can do. You could try to limit your items, or selectively remove items (especially highly correlated ones) to see if that helps. Collecting more data can help too. There are some mathematical fudges you can do, but they're not as tasty as vanilla fudge and they are hard to implement easily.

Executing this command creates a matrix of correlation coefficients called *raqMatrix*. We can use this matrix in the analysis (although we don't have to). It's a good idea to have a look at the correlation matrix, for the reasons we discussed earlier. To make our eyes hurt a little less, let's use the *round()* function to display only 2 decimal places of the correlation matrix that we have just created:

```
round(raqMatrix, 2)
```

The *R*-matrix (or correlation matrix) produced using the *cor()* function is displayed in Output 17.1. You should be comfortable with the idea that to do a factor analysis we need to have variables that correlate fairly well, but not perfectly. Also, any variables that correlate with no others should be eliminated. Therefore, we can use this correlation matrix to check the pattern of relationships. First, scan the matrix for correlations greater than .3, then look for variables that only have a small number of correlations greater than this value. Then scan the correlation coefficients themselves and look for any greater than .9. If any are found then you should be aware that a problem could arise because of multicollinearity in the data.

```
      Q01   Q02   Q03   Q04   Q05   Q06   Q07   Q08
Q01  1.00 -0.10 -0.34  0.44  0.40  0.22  0.31  0.33
Q02 -0.10  1.00  0.32 -0.11 -0.12 -0.07 -0.16 -0.05
Q03 -0.34  0.32  1.00 -0.38 -0.31 -0.23 -0.38 -0.26
Q04  0.44 -0.11 -0.38  1.00  0.40  0.28  0.41  0.35
Q05  0.40 -0.12 -0.31  0.40  1.00  0.26  0.34  0.27
Q06  0.22 -0.07 -0.23  0.28  0.26  1.00  0.51  0.22
Q07  0.31 -0.16 -0.38  0.41  0.34  0.51  1.00  0.30
Q08  0.33 -0.05 -0.26  0.35  0.27  0.22  0.30  1.00
Q09 -0.09  0.31  0.30 -0.12 -0.10 -0.11 -0.13  0.02
Q10  0.21 -0.08 -0.19  0.22  0.26  0.32  0.28  0.16
Q11  0.36 -0.14 -0.35  0.37  0.30  0.33  0.34  0.63
Q12  0.35 -0.19 -0.41  0.44  0.35  0.31  0.42  0.25
Q13  0.35 -0.14 -0.32  0.34  0.30  0.47  0.44  0.31
Q14  0.34 -0.16 -0.37  0.35  0.32  0.40  0.44  0.28
Q15  0.25 -0.16 -0.31  0.33  0.26  0.36  0.39  0.30
Q16  0.50 -0.17 -0.42  0.42  0.39  0.24  0.39  0.32
Q17  0.37 -0.09 -0.33  0.38  0.31  0.28  0.39  0.59
Q18  0.35 -0.16 -0.38  0.38  0.32  0.51  0.50  0.28
Q19 -0.19  0.20  0.34 -0.19 -0.17 -0.17 -0.27 -0.16
Q20  0.21 -0.20 -0.32  0.24  0.20  0.10  0.22  0.18
Q21  0.33 -0.20 -0.42  0.41  0.33  0.27  0.48  0.30
Q22 -0.10  0.23  0.20 -0.10 -0.13 -0.17 -0.17 -0.08
Q23  0.00  0.10  0.15 -0.03 -0.04 -0.07 -0.07 -0.05

      Q09   Q10   Q11   Q12   Q13   Q14   Q15   Q16
Q01 -0.09  0.21  0.36  0.35  0.35  0.34  0.25  0.50
Q02  0.31 -0.08 -0.14 -0.19 -0.14 -0.16 -0.16 -0.17
Q03  0.30 -0.19 -0.35 -0.41 -0.32 -0.37 -0.31 -0.42
Q04 -0.12  0.22  0.37  0.44  0.34  0.35  0.33  0.42
Q05 -0.10  0.26  0.30  0.35  0.30  0.32  0.26  0.39
Q06 -0.11  0.32  0.33  0.31  0.47  0.40  0.36  0.24
Q07 -0.13  0.28  0.34  0.42  0.44  0.44  0.39  0.39
Q08  0.02  0.16  0.63  0.25  0.31  0.28  0.30  0.32
Q09  1.00 -0.13 -0.12 -0.17 -0.17 -0.12 -0.19 -0.19
Q10 -0.13  1.00  0.27  0.25  0.30  0.25  0.30  0.29
Q11 -0.12  0.27  1.00  0.34  0.42  0.33  0.36  0.37
Q12 -0.17  0.25  0.34  1.00  0.49  0.43  0.33  0.41
Q13 -0.17  0.30  0.42  0.49  1.00  0.45  0.34  0.36
```

```
Q14 -0.12  0.25  0.33  0.43  0.45  1.00  0.38  0.42
Q15 -0.19  0.30  0.36  0.33  0.34  0.38  1.00  0.45
Q16 -0.19  0.29  0.37  0.41  0.36  0.42  0.45  1.00
Q17 -0.04  0.22  0.59  0.33  0.41  0.35  0.37  0.41
Q18 -0.15  0.29  0.37  0.49  0.53  0.50  0.34  0.42
Q19  0.25 -0.13 -0.20 -0.27 -0.23 -0.25 -0.21 -0.27
Q20 -0.16  0.08  0.26  0.30  0.20  0.23  0.21  0.27

      Q17   Q18   Q19   Q20   Q21   Q22   Q23
Q01  0.37  0.35 -0.19  0.21  0.33 -0.10  0.00
Q02 -0.09 -0.16  0.20 -0.20 -0.20  0.23  0.10
Q03 -0.33 -0.38  0.34 -0.32 -0.42  0.20  0.15
Q04  0.38  0.38 -0.19  0.24  0.41 -0.10 -0.03
Q05  0.31  0.32 -0.17  0.20  0.33 -0.13 -0.04
Q06  0.28  0.51 -0.17  0.10  0.27 -0.17 -0.07
Q07  0.39  0.50 -0.27  0.22  0.48 -0.17 -0.07
Q08  0.59  0.28 -0.16  0.18  0.30 -0.08 -0.05
Q09 -0.04 -0.15  0.25 -0.16 -0.14  0.26  0.17
Q10  0.22  0.29 -0.13  0.08  0.19 -0.13 -0.06
Q11  0.59  0.37 -0.20  0.26  0.35 -0.16 -0.09
Q12  0.33  0.49 -0.27  0.30  0.44 -0.17 -0.05
Q13  0.41  0.53 -0.23  0.20  0.37 -0.20 -0.05
Q14  0.35  0.50 -0.25  0.23  0.40 -0.17 -0.05
Q15  0.37  0.34 -0.21  0.21  0.30 -0.17 -0.06
Q16  0.41  0.42 -0.27  0.27  0.42 -0.16 -0.08
Q17  1.00  0.38 -0.16  0.21  0.36 -0.13 -0.09
Q18  0.38  1.00 -0.26  0.24  0.43 -0.16 -0.08
Q19 -0.16 -0.26  1.00 -0.25 -0.27  0.23  0.12
Q20  0.21  0.24 -0.25  1.00  0.47 -0.10 -0.03
Q21  0.36  0.43 -0.27  0.47  1.00 -0.13 -0.07
Q22 -0.13 -0.16  0.23 -0.10 -0.13  1.00  0.23
Q23 -0.09 -0.08  0.12 -0.03 -0.07  0.23  1.00
```

Output 17.1

As well as looking at the correlation matrix, we should run Bartlett's test and the KMO on the correlation matrix. Bartlett's test is run using the **cortest.bartlett()** function from the *psych* package. We can run this test either on the raw data or on the correlation matrix. To run it from the raw data simply input the dataframe (in this case *raqData*) into the function:

```
cortest.bartlett(raqData)
```

To run it from the correlation matrix (in this case *raqMatrix*), input the name of the correlation matrix but also provide the sample size (in this case 2751):

```
cortest.bartlett(raqMatrix, n = 2571)
```

Both methods will give you the results in Output 17.2. If you ran the test from the raw data, you'll get the warning *R was not square, finding R from data*, which is nothing to worry about it just means that because we didn't give the function a correlation matrix, it's calculating it from the raw data (that's what we expect it to do). For factor analysis to work we need some relationships between variables and if the *R*-matrix were an identity matrix then all correlation coefficients would be zero. Therefore, we want this test to be *significant* (i.e., have a significance value less than .05). A significant test tells us that the *R*-matrix is not an identity matrix; therefore, there are some relationships between the variables we hope to include in the analysis. For these data, Bartlett's test is highly significant, $\chi^2(253) = 19{,}334$, $p < .001$, and therefore factor analysis is appropriate.

```
R was not square, finding R from data
$chisq
[1] 19334.49

$p.value
[1] 0

$df
[1] 253
```

Output 17.2

Next we'd also like the KMO. None of the packages in R currently have a straightforward way to calculate the KMO. However, one of the nice things about R is that people can write programs to do anything that R doesn't currently do, and G. Jay Kerns, from Youngstown State University (see http://tolstoy.newcastle.edu.au/R/e2/help/07/08/22816.html) has written one called **kmo()**, which calculates the KMO and a variety of other things. The function itself is easy to use manually (see Oliver Twisted), but because it is not part of a package we have included it in our DSUR package so that you can use it directly (assuming you have loaded the DSUR package). You can use the function by simply entering the name of your dataframe into it and executing.

```
kmo(raqData)
```

The results of the KMO test are shown in Output 17.3. We came across the KMO statistic in section 17.4.1 and saw that Kaiser (1974) recommends a bare minimum of .5 and that values between .5 and .7 are mediocre, values between .7 and .8 are good, values between .8 and .9 are great and values above .9 are superb (Hutcheson & Sofroniou, 1999). For these data the overall value is .93, which falls into the range of being superb (or 'marvellous' as the report puts it), so we should be confident that the sample size and the data are adequate for factor analysis.

OLIVER TWISTED

Please Sir, can I have some more ... kmo?

'Stop spanking my monkey!', cries an hysterical Oliver, 'it's never done you any harm, and it's orange.' I was talking about the Kaiser–Meyer–Olkin test, Oliver. 'Oh, sorry', he says with a sigh of relief, 'I thought KMO stood for Kill My Orang-utan'. Erm, OK, Oliver has finally lost the plot, which I'm fairly sure is what you'll do if you inspect the *kmo()* function on the companion website. Although we have included it in our DSUR package, you can also copy it and execute it manually.

KMO can be calculated for multiple and individual variables. The value of KMO should be above the bare minimum of .5 for all variables (and preferably higher) as well as overall. The KMO values for individual variables are produced by the *kmo()* function too. For these data all values are well above .5, which is good news. If you find any variables with values below .5 then you should consider excluding them from the analysis (or run the analysis with and without that variable and note the difference). Removal of a variable affects the KMO statistics, so if you do remove a variable be sure to rerun the *kmo()* function on the new data.

```
$overall
[1] 0.9302245

$report
[1] "The KMO test yields a degree of common variance marvelous."

$individual

Q01      Q02      Q03      Q04      Q05      Q06      Q07
0.9297   0.8748   0.9510   0.9553   0.9601   0.8913   0.9417

Q08      Q09      Q10      Q11      Q12      Q13      Q14
0.8713   0.8337   0.9487   0.9059   0.9548   0.9482   0.9672

Q15      Q16      Q17      Q18      Q19      Q20      Q21
0.9404   0.9336   0.9306   0.9479   0.9407   0.8891   0.9293

Q22      Q23
0.8784   0.7664
```

Output 17.3

Finally, we'd like the determinant of the correlation matrix. To find the determinant, we use the **det()** function, into which we place the name of a correlation matrix. We have computed this matrix already for the current data (*raqMatrix*) so we can execute:

```
det(raqMatrix)
```

If we hadn't already created the matrix, we could get the determinant by putting the *cor()* function for the raw data into the *det()* function:

```
det(cor(raqData))
```

Either method produces the same value:

```
[1] 0.0005271037
```

This value is greater than the necessary value of 0.00001 (see section 17.5). As such, our determinant does not seem problematic. After checking the determinant, you can, if necessary, eliminate variables that you think are causing the problem. In summary, all questions in the RAQ correlate reasonably well with all others and none of the correlation coefficients are excessively large; therefore, we won't eliminate any questions at this stage.

CRAMMING SAM'S TIPS Preliminary analysis

- Scan the *correlation matrix*; look for variables that don't correlate with any other variables, or correlate very highly ($r = .9$) with one or more other variables. In factor analysis, check that the determinant of this matrix is bigger than 0.00001; if it is then multicollinearity isn't a problem.
- Check the *KMO and Bartlett's test*; the KMO statistic should be greater than .5 as a bare minimum; if it isn't collect more data. Bartlett's test of sphericity should be significant (the significance value should be less than .05).

17.6.3. Factor extraction using R ②

For our present purposes we will use *principal components analysis*, which strictly speaking isn't factor analysis; however, the two procedures may often yield similar results (see section 17.3.6). Principal component analysis is carried out using the **principal()** function, in the *psych* package. This function takes the general form:

```
pcModel<-principal(dataframe/R-matrix, nfactors = number of factors, rotate =
"method of rotation", scores = TRUE/FALSE)
```

This command creates a principal components model called *pcModel*, by specifying either a dataframe of raw data or a correlation matrix. There are three main options:

- *nfactors* allows you to specify how many factors/components you want to extract (see section 17.3.8) as a number. If you don't specify *nfactors*, then one component is extracted.
- *rotate* allows you to specify a method of factor rotation (see section 17.3.9) using a text string. If you don't declare a method of rotation, the default of varimax rotation is used.
- *scores* allows you to obtain factor scores (TRUE) or not (FALSE). The default is FALSE.

I mentioned earlier that when conducting principal components analysis we begin by establishing the linear variates within the data and then decide how many of these variates to retain (or 'extract'). Therefore, our starting point is to create a principal components model that has the same number of factors as there are variables in the data: by doing this we are just reducing the data set down to its underlying factors. By extracting as many factors as there are variables we can inspect their eigenvalues and make decisions about which factors to extract. (Note that if you use factor analysis, rather than principal components analysis, you need to extract fewer factors than you have variables – so if you have 23 variables, extracting 18, or so, factors should be OK.)

To create this model we execute one of these commands:

```
pc1 <-  principal(raqData, nfactors = 23, rotate = "none")
pc1 <- principal(raqMatrix, nfactors = 23, rotate = "none")
```

The first command creates the model from the raw data and the second from the correlation matrix: both methods will give you identical results, but we will show both throughout. These commands create a model called *pc1*, which extracts 23 factors – the same as

R's Souls' Tip 17.2 A cure for lazy-itis ②

Sometimes, I'm too lazy to count the variables in my data set, in which case I can ask **R** to count them for me, using the *length()* function, which counts the number of items in an object. Therefore, we can obtain the number of variables in a dataframe using:

```
length(dataFrame)
```

Similarly, we can apply this function to a matrix to find out the number of rows in a column of a matrix:

```
length(matrix[,1])
```

Therefore, we can use these commands within the *principal()* function to automatically specify the number of factors as the number of variables in the dataframe/matrix by executing:

```
pc2 <-  principal(raq, nfactors=length(raqData), rotate="none")
pc2 <- principal(raqmatrix, nfactors=length(raqMatrix[,1]), rotate="none")
```

the number of variables. If you have a large data set or are just too lazy to remember how many variables you have then you can change the command slightly to get **R** to calculate the number of variables in the dataframe or correlation matrix automatically (see R's Souls' Tip 17.2). A final thing to note is that we have set the rotation method to *"none"*, which means that we won't carry out factor rotation because we don't need to at this stage.

We can look at the results of the principal components analysis by executing its name:

```
pc1
```

Output 17.4 shows the results of the first principal components model. The first part of this is the unrotated loadings. Currently these are not interesting, but they represent the loading from each factor or component to each variable.

```
Principal Components Analysis
Call: principal(r = raq, nfactors = 23, rotate = "none")
Standardized loadings based upon correlation matrix
        PC1   PC2   PC3   PC4   PC5   PC6   PC7   PC8
Q01    0.59  0.18 -0.22  0.12 -0.40 -0.11 -0.22 -0.08
Q02   -0.30  0.55  0.15  0.01 -0.03 -0.38  0.19 -0.39
Q03   -0.63  0.29  0.21 -0.07  0.02  0.00  0.01 -0.05
Q04    0.63  0.14 -0.15  0.15 -0.20 -0.12 -0.06  0.11
Q05    0.56  0.10 -0.07  0.14 -0.42 -0.17 -0.06  0.11
Q06    0.56  0.10  0.57 -0.05  0.17  0.01  0.00  0.05
Q07    0.69  0.04  0.25  0.10  0.17 -0.08  0.05  0.03
Q08    0.55  0.40 -0.32 -0.42  0.15  0.10 -0.07 -0.04
Q09   -0.28  0.63 -0.01  0.10  0.17 -0.27 -0.01 -0.03
Q10    0.44  0.03  0.36 -0.10 -0.34  0.22  0.44 -0.03
Q11    0.65  0.25 -0.21 -0.40  0.13  0.18 -0.01  0.03
Q12    0.67 -0.05  0.05  0.25  0.04 -0.08 -0.14  0.08
Q13    0.67  0.08  0.28 -0.01  0.13  0.03 -0.21  0.05
Q14    0.66  0.02  0.20  0.14  0.08 -0.03 -0.10 -0.06
Q15    0.59  0.01  0.12 -0.11 -0.07  0.29  0.32 -0.12
Q16    0.68  0.01 -0.14  0.08 -0.32  0.00  0.12 -0.14
Q17    0.64  0.33 -0.21 -0.34  0.10  0.05 -0.02  0.03
Q18    0.70  0.03  0.30  0.13  0.15 -0.09 -0.10  0.06
Q19   -0.43  0.39  0.10 -0.01 -0.15  0.07  0.05  0.68
Q20    0.44 -0.21 -0.40  0.30  0.33 -0.01  0.34  0.03
Q21    0.66 -0.06 -0.19  0.28  0.24 -0.15  0.18  0.10
Q22   -0.30  0.47 -0.12  0.38  0.07  0.12  0.31  0.12
Q23   -0.14  0.37 -0.02  0.51  0.02  0.62 -0.28 -0.22

...

        PC17  PC18  PC19  PC20  PC21  PC22  PC23 h2       u2
Q01    -0.05 -0.17  0.16 -0.01 -0.21  0.05  0.01  1  0.0e+00
Q02    -0.08  0.00  0.01 -0.02 -0.02  0.03  0.02  1 -3.1e-15
Q03     0.43  0.08  0.09  0.05  0.01  0.00  0.05  1 -1.6e-15
Q04     0.19  0.05 -0.21  0.04  0.09 -0.02  0.02  1 -1.1e-15
Q05    -0.04  0.01 -0.04  0.00 -0.02  0.02  0.01  1 -2.0e-15
Q06    -0.14  0.05  0.09 -0.07  0.04 -0.32 -0.11  1  0.0e+00
Q07     0.03 -0.15  0.20  0.16  0.14  0.24  0.09  1  1.1e-16
Q08     0.10  0.07  0.12 -0.15  0.06  0.16 -0.36  1 -2.2e-16
Q09    -0.19 -0.02 -0.08 -0.03  0.04 -0.01  0.03  1 -4.4e-16
Q10     0.07 -0.01  0.00  0.04 -0.03  0.02 -0.04  1 -4.4e-16
Q11    -0.05  0.07  0.07 -0.18  0.06  0.00  0.41  1 -8.9e-16
Q12    -0.08  0.04  0.36  0.00 -0.04 -0.10 -0.02  1 -2.2e-16
Q13    -0.06 -0.32 -0.30 -0.06  0.16  0.08 -0.05  1  0.0e+00
Q14     0.34 -0.09  0.06  0.02  0.03 -0.01  0.05  1 -4.4e-16
Q15    -0.12 -0.10 -0.04 -0.07 -0.19  0.10  0.00  1 -4.4e-16
Q16    -0.03  0.22 -0.02 -0.04  0.35 -0.12 -0.01  1 -2.0e-15
```

```
Q17          0.04 -0.04 -0.10  0.42 -0.15 -0.23 -0.01  1 -4.4e-16
Q18         -0.06  0.45 -0.15  0.08 -0.18  0.23  0.01  1 -8.9e-16
Q19         -0.06  0.01  0.05 -0.02  0.02  0.04 -0.02  1 -6.7e-16
Q20         -0.09  0.00  0.04  0.18  0.10  0.06 -0.04  1 -8.9e-16
Q21          0.20 -0.03 -0.11 -0.31 -0.20 -0.13 -0.01  1 -2.0e-15
Q22          0.04 -0.06  0.02  0.00  0.01 -0.01  0.01  1  0.0e+00
Q23         -0.03  0.05 -0.03  0.01 -0.01 -0.02  0.00  1  0.0e+00

                PC1  PC2  PC3  PC4  PC5  PC6  PC7  PC8  PC9
SS loadings    7.29 1.74 1.32 1.23 0.99 0.90 0.81 0.78 0.75
Proportion Var 0.32 0.08 0.06 0.05 0.04 0.04 0.04 0.03 0.03
Cumulative Var 0.32 0.39 0.45 0.50 0.55 0.59 0.62 0.65 0.69

               PC10 PC11 PC12 PC13 PC14 PC15 PC16 PC17 PC18
SS loadings    0.72 0.68 0.67 0.61 0.58 0.55 0.52 0.51 0.46
Proportion Var 0.03 0.03 0.03 0.03 0.03 0.02 0.02 0.02 0.02
Cumulative Var 0.72 0.75 0.78 0.80 0.83 0.85 0.88 0.90 0.92

               PC19 PC20 PC21 PC22 PC23
SS loadings    0.42 0.41 0.38 0.36 0.33
Proportion Var 0.02 0.02 0.02 0.02 0.01
Cumulative Var 0.94 0.95 0.97 0.99 1.00

Test of the hypothesis that 23 factors are sufficient.

The degrees of freedom for the null model are 253 and the objective
function was 7.55
The degrees of freedom for the model are -23  and the objective
function was 0
The number of observations was 2571 with Chi Square =  0 with prob < NA

Fit based upon off diagonal values = 1
```

Output 17.4

On the far right of the factor loading matrix are two columns, labelled *h2* and *u2*. *h2* is the communalities (which are sometimes called h^2). These communalities are all equal to 1 because we have extracted 23 items, the same as the number of variables: we've explained all of the variance in every variable. When we extract fewer factors (or components) we'll have lower communalities. Next to the communality column is the uniqueness column, labelled *u2*. This is the amount of unique variance for each variable, and it's 1 minus the communality; because all of the communalities are 1, all of the uniquenesses are 0.[8]

The next thing to look at after the factor loading matrix is the eigenvalues. The eigenvalues associated with each factor represent the variance explained by that particular linear component. **R** calls these SS loadings (sums of squared loadings), because they are the sum of the squared loadings. (You can also find them in a variable associated with the model called *values*, so in our case we could access this variable using *pc1$values*).

R also displays the eigenvalues in terms of the proportion of variance explained. Factor 1 explains 7.29 units of variance out of a possible 23 (the number of factors) so as a proportion this is 7.29/23 = 0.32; this is the value that **R** reports. We can convert these proportions to percentages by multiplying by 100; so, factor 1 explains 32% of the total variance.

[8] Some of them are very, very slightly different from zero; for example, question 2 has a uniqueness, which is reported as −3.1e-15, which means .0000000000000031. This is caused by a rounding error (because **R** stores variables to only 15 decimal places).

It should be clear that the first few factors explain relatively large amounts of variance (especially factor 1) whereas subsequent factors explain only small amounts of variance.

The eigenvalues show us that four components (or factors) have eigenvalues greater than 1, suggesting that we extract four components if we use Kaiser's criterion. By Jolliffe's criterion (retain factors with eigenvalues greater than 0.7) we should retain 10 factors, but there is little to recommend this criterion over Kaiser's. We should also consider the scree plot. As mentioned above, the eigenvalues are stored in a variable called *pc1$values*, and we can draw a quick scree plot using the *plot()* function, by executing:

```
plot(pc1$values, type = "b")
```

This command simply plots the eigenvalues (*y*) against the factor number (*x*). By default, the *plot()* function will plot points (*type= "p"*). We want to see a line so that we can look at the trend (we could ask for this by specifying *type="l"*), but ideally we want to look at both a line and points on the same graph, which is why we specify *type="b"*.

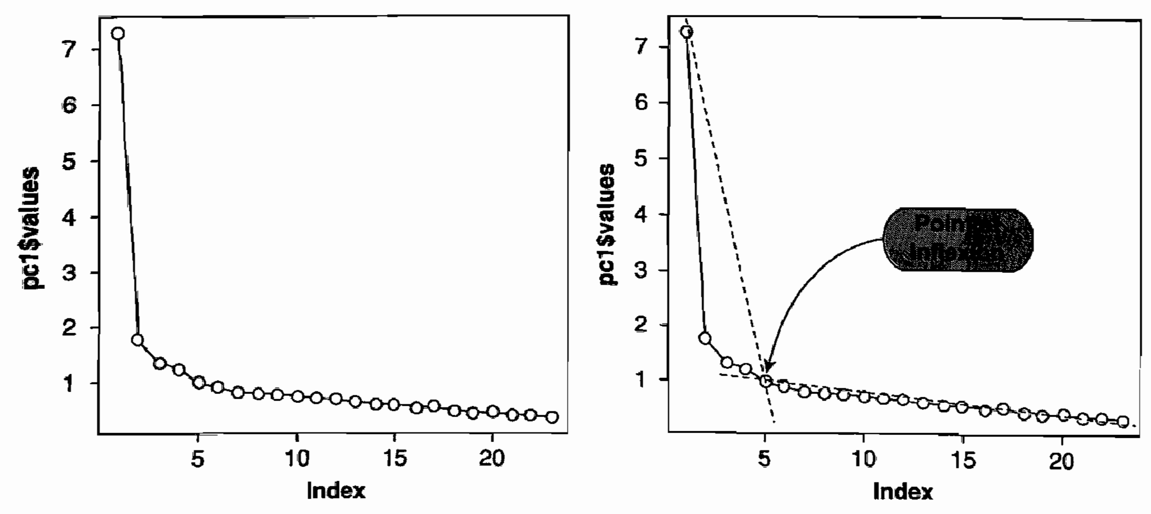


FIGURE 17.7 Scree plot from principal components analysis of RAQ data. The second plot shows the point of inflexion at the fourth component.

Figure 17.7 shows the scree plot; I show it once as R produces it and then again with lines showing a plateau and (what I consider to be) the point of inflexion. This curve is difficult to interpret because it begins to tail off after three factors, but there is another drop after four factors before a stable plateau is reached. Therefore, we could probably justify retaining either two or four factors. Given the large sample, it is probably safe to assume Kaiser's criterion. The evidence from the scree plot and from the eigenvalues suggests a four-component solution may be the best.

Now that we know how many components we want to extract, we can rerun the analysis, specifying that number. To do this, we use an identical command to the previous model but we change *nfactors = 23* to be *nfactors = 4* because we now want only four factors. (We should also change the name of the resulting model so that we don't overwrite the previous one):

```
pc2 <-  principal(raqData, nfactors = 4, rotate = "none")
pc2 <- principal(raqMatrix, nfactors = 4, rotate = "none")
```

Again, the first command is to run the analysis from the raw data and the second is if you're using the correlation matrix. In both cases the commands create a model called *pc2* that is the same as before except that we've extracted only 4 factors (not 23). We can look at this model by executing its name:

```
pc2
```

Output 17.5 shows the second principal components model. Again, the output contains the unrotated factor loadings, but only for the first four factors. Notice that these are unchanged from the previous factor loading matrix. Also notice that the eigenvalues (SS loadings), proportions of variance explained and cumulative proportion of variance explained are also unchanged (except now there are only four of them, because we only have four components). However, the communalities (the *h*2 column) and uniquenesses (the *u*2 column) are changed. Remember that the communality is the proportion of common variance within a variable (see section 17.3.4). Principal components analysis works on the initial assumption that all variance is common; therefore, before extraction the communalities are all 1. In effect, all of the variance associated with a variable is assumed to be common variance. Once factors have been extracted, we have a better idea of how much variance is, in reality, common. The communalities in the output reflect this common variance. So, for example, we can say that 43% of the variance associated with question 1 is common, or shared, variance. Another way to look at these communalities is in terms of the proportion of variance explained by the underlying factors. Before extraction, there were as many factors as there are variables, so all variance is explained by the factors and communalities are all 1. However, after extraction some of the factors are discarded and so some information is lost. The retained factors cannot explain all of the variance present in the data, but they can explain some. The amount of variance in each variable that can be explained by the retained factors is represented by the communalities after extraction.

Now that we have the communalities, we can go back to Kaiser's criterion to see whether we still think that four factors should have been extracted. In section 17.3.8 we saw that Kaiser's criterion is accurate when there are fewer than 30 variables and communalities after extraction are greater than .7 or when the sample size exceeds 250 and the average communality is greater than .6. Of the communalities in Output 17.5, only one exceeds .7. The average of these communalities can be found by adding them up and dividing by the number of communalities (11.573/23 = .503). So, on both grounds Kaiser's rule may not be accurate. However, in this instance we should consider the huge sample that we have, because the research into Kaiser's criterion gives recommendations for much smaller samples. It's also worth remembering that we have already inspected the scree plot, which should be a good guide in a sample as large as ours. However, given the ambiguity in the scree plot (there was also a case for retaining only two factors) you might like to rerun the analysis specifying that R extract only two factors and compare the results.

```
Principal Components Analysis
Call: principal(r = raq, nfactors = 4, rotate = "none")
Standardized loadings based upon correlation matrix
      PC1   PC2   PC3   PC4   h2   u2
Q01  0.59  0.18 -0.22  0.12 0.43 0.57
Q02 -0.30  0.55  0.15  0.01 0.41 0.59
Q03 -0.63  0.29  0.21 -0.07 0.53 0.47
Q04  0.63  0.14 -0.15  0.15 0.47 0.53
Q05  0.56  0.10 -0.07  0.14 0.34 0.66
Q06  0.56  0.10  0.57 -0.05 0.65 0.35
Q07  0.69  0.04  0.25  0.10 0.55 0.45
Q08  0.55  0.40 -0.32 -0.42 0.74 0.26
Q09 -0.28  0.63 -0.01  0.10 0.48 0.52
Q10  0.44  0.03  0.36 -0.10 0.33 0.67
Q11  0.65  0.25 -0.21 -0.40 0.69 0.31
Q12  0.67 -0.05  0.05  0.25 0.51 0.49
Q13  0.67  0.08  0.28 -0.01 0.54 0.46
Q14  0.66  0.02  0.20  0.14 0.49 0.51
Q15  0.59  0.01  0.12 -0.11 0.38 0.62
Q16  0.68  0.01 -0.14  0.08 0.49 0.51
Q17  0.64  0.33 -0.21 -0.34 0.68 0.32
```

```
Q18  0.70  0.03  0.30  0.13 0.60 0.40
Q19 -0.43  0.39  0.10 -0.01 0.34 0.66
Q20  0.44 -0.21 -0.40  0.30 0.48 0.52
Q21  0.66 -0.06 -0.19  0.28 0.55 0.45
Q22 -0.30  0.47 -0.12  0.38 0.46 0.54
Q23 -0.14  0.37 -0.02  0.51 0.41 0.59

                 PC1  PC2  PC3  PC4
SS loadings     7.29 1.74 1.32 1.23
Proportion Var 0.32 0.08 0.06 0.05
Cumulative Var 0.32 0.39 0.45 0.50

Test of the hypothesis that 4 factors are sufficient.

The degrees of freedom for the null model are 253 and the objective
function was 7.55
The degrees of freedom for the model are 167  and the objective
function was 1.03
The number of observations was 2571 with Chi Square = 2634.37 with prob
< 0

Fit based upon off diagonal values = 0.96
```

Output 17.5

There's another thing that we can look at to see if we've extracted the correct number of factors: this is the reproduced correlation matrix and the difference between the reproduced correlation matrix and the correlation matrix in the data.

The reproduced correlations are obtained with the **factor.model()** function. The *factor.model()* function, needs to know the factor loading matrix. The factor loading matrix is labelled as an object called *loadings* in the principal components model; therefore we can access it by specifying *pc2$loadings* (which translates as 'the *loadings* object associated with the *pc2* model). Therefore, we can get the reproduced correlations by executing:

```
factor.model(pc2$loadings)
```

The difference between the reproduced and actual correlation matrices is referred to as the residuals, and these are obtained with the **factor.residuals()** function. You again need to provide the factor loading matrix but also the correlation matrix to which you want to compare it (in this case the original correlation matrix, *raqMatrix*). We can, therefore, obtain the residuals by executing:

```
factor.residuals(raqMatrix, pc2$loadings)
        Q01    Q02    Q03    Q04    Q05    Q06    Q07    Q08    Q09
Q01  0.435 -0.112 -0.372  0.447  0.376  0.218  0.366  0.412 -0.042
Q02 -0.112  0.414  0.380 -0.134 -0.122 -0.033 -0.148  0.002  0.430
Q03 -0.372  0.380  0.530 -0.399 -0.345 -0.200 -0.373 -0.270  0.352
Q04  0.447 -0.134 -0.399  0.469  0.399  0.278  0.419  0.390 -0.073
Q05  0.376 -0.122 -0.345  0.399  0.343  0.273  0.380  0.312 -0.080
Q06  0.218 -0.033 -0.200  0.278  0.273  0.654  0.528  0.183 -0.108
Q07  0.366 -0.148 -0.373  0.419  0.380  0.528  0.545  0.267 -0.161
Q08  0.412  0.002 -0.270  0.390  0.312  0.183  0.267  0.739  0.055
Q09 -0.042  0.430  0.352 -0.073 -0.080 -0.108 -0.161  0.055  0.484
Q10  0.172 -0.061 -0.181  0.212  0.205  0.461  0.382  0.180 -0.116
Q11  0.423 -0.097 -0.357  0.419  0.348  0.290  0.363  0.691 -0.071
Q12  0.402 -0.219 -0.440  0.448  0.397  0.388  0.495  0.228 -0.195
Q13  0.347 -0.122 -0.342  0.395  0.360  0.545  0.533  0.313 -0.147
Q14  0.362 -0.155 -0.373  0.411  0.370  0.477  0.514  0.249 -0.159
```

```
Q15  0.311 -0.158 -0.337  0.343  0.306  0.406  0.425  0.339 -0.174
Q16  0.440 -0.217 -0.458  0.466  0.400  0.300  0.439  0.390 -0.175
Q17  0.439 -0.048 -0.331  0.434  0.359  0.290  0.365  0.695 -0.009
Q18  0.368 -0.149 -0.376  0.424  0.388  0.562  0.570  0.250 -0.168
Q19 -0.204  0.357  0.403 -0.231 -0.207 -0.147 -0.254 -0.104  0.363
Q20  0.342 -0.301 -0.440  0.353  0.292 -0.021  0.219  0.164 -0.218
Q21  0.449 -0.254 -0.488  0.480  0.412  0.244  0.430  0.282 -0.191
Q22 -0.025  0.333  0.275 -0.050 -0.060 -0.209 -0.179 -0.099  0.417
Q23  0.045  0.246  0.158  0.042  0.028 -0.082 -0.037 -0.136  0.323
```

Output 17.6

Output 17.6 shows an edited version of the reproduced correlation matrix that was requested using the *factor.model()* function in the first table. The diagonal of this matrix contains the communalities after extraction for each variable (you can check the values against Output 17.5). Output 17.7 contains an extract from the matrix of residuals: the difference between the fitted model and the real data. The diagonal of this matrix is the uniquenesses.

```
       Q01    Q02    Q03    Q04    Q05    Q06    Q07    Q08    Q09
Q01  0.565  0.013  0.035 -0.011  0.027 -0.001 -0.061 -0.081 -0.050
Q02  0.013  0.586 -0.062  0.022  0.003 -0.041 -0.011 -0.052 -0.115
Q03  0.035 -0.062  0.470  0.019  0.035 -0.027 -0.009  0.011 -0.052
Q04 -0.011  0.022  0.019  0.531  0.002  0.000 -0.010 -0.041 -0.051
Q05  0.027  0.003  0.035  0.002  0.657 -0.016 -0.041 -0.044 -0.016
Q06 -0.001 -0.041 -0.027  0.000 -0.016  0.346 -0.014  0.040 -0.005
Q07 -0.061 -0.011 -0.009 -0.010 -0.041 -0.014  0.455  0.030  0.033
Q08 -0.081 -0.052  0.011 -0.041 -0.044  0.040  0.030  0.261 -0.039
Q09 -0.050 -0.115 -0.052 -0.051 -0.016 -0.005  0.033 -0.039  0.516
Q10  0.042 -0.023 -0.013  0.003  0.053 -0.139 -0.098 -0.021 -0.018
Q11 -0.066 -0.046  0.006 -0.051 -0.050  0.038 -0.018 -0.061 -0.045
Q12 -0.057  0.024  0.030 -0.006 -0.050 -0.076 -0.072  0.024  0.027
Q13  0.008 -0.021  0.024 -0.051 -0.058 -0.078 -0.091  0.001 -0.021
Q14 -0.024 -0.009  0.002 -0.060 -0.055 -0.075 -0.074  0.032  0.038
Q15 -0.065 -0.007  0.025 -0.009 -0.045 -0.047 -0.033 -0.039 -0.012
Q16  0.059  0.050  0.039 -0.050 -0.005 -0.056 -0.051 -0.068 -0.014
Q17 -0.069 -0.039  0.003 -0.052 -0.049 -0.008  0.025 -0.105 -0.027
Q18 -0.020 -0.015  0.001 -0.042 -0.066 -0.048 -0.069  0.030  0.018
Q19  0.015 -0.153 -0.061  0.045  0.041 -0.020 -0.015 -0.056 -0.114
Q20 -0.128  0.099  0.115 -0.110 -0.092  0.122  0.002  0.011  0.060
Q21 -0.120  0.049  0.071 -0.070 -0.078  0.029  0.053  0.014  0.055
Q22 -0.079 -0.102 -0.071 -0.049 -0.072  0.043  0.010  0.020 -0.161
Q23 -0.049 -0.147 -0.008 -0.076 -0.070  0.013 -0.033  0.086 -0.152
```

Output 17.7

The correlations in the reproduced matrix differ from those in the *R*-matrix because they stem from the model rather than the observed data. If the model were a perfect fit to the data then we would expect the reproduced correlation coefficients to be the same as the original correlation coefficients. Therefore, to assess the fit of the model we can look at the differences between the observed correlations and the correlations based on the model. For example, if we take the correlation between questions 1 and 2, the correlation based on the observed data is −.099 (taken from Output 17.1). The correlation based on the model is −.112, which is slightly higher. We can calculate the difference as follows :

$$\begin{aligned}\text{residual} &= r_{\text{observed}} - r_{\text{from model}}\\ \text{residual}_{Q_1Q_2} &= (-0.099) - (-0.112)\\ &= 0.013\end{aligned}$$

You should notice that this difference is the value quoted in Output 17.7 for questions 1 and 2. Therefore, Output 17.7 contains the differences between the observed correlation coefficients and the ones predicted from the model. For a good model these values will all be small. There are several ways we can define how small we want the residuals to be.

One approach is to see how large the residuals are, compared to the original correlations. The very worst the model could be (if we extracted no factors at all) would be the size of the correlations in the original data. Thus one approach is to compare the size of the residuals with the size of the correlations. If the correlations were small to start with, we'd expect very small residuals. If the correlations were large to start with, we wouldn't mind if the residuals were relatively larger. So one measure of the residuals is to compare the residuals with the original correlations – because residuals are positive and negative, they should be squared before doing that. A measure of the fit of the model is therefore the sum of the squared residuals divided by the sum of the squared correlations. As this is considered a measure of fit and sometimes people like measures of fit to go from 0 to 1, we subtract the value from 1. This statistic is given at the bottom of the main output (Output 17.5) as:

```
Fit based upon off diagonal values = 0.96
```

Values over 0.95 are often considered indicators of good fit, and as our value is 0.96, this indicates that four factors are sufficient.

There are many other ways of looking at residuals, which we'll now explore. We couldn't find an **R** function to do these other things, but we will write one as we go along.[9] A simple approach to residuals is just to say that we want the residuals to be small. In fact, we want most values to be less than 0.05. We can work out how many residuals are large by this criterion fairly easily in **R**. First, we need to extract the residuals into a new object. We need to do this because at the moment the matrix of residuals is symmetrical (so the residuals are repeated above and below the diagonal of the matrix), and also the diagonal of the matrix does not contain residuals. First let's create an object called *residuals* that contains the factor residuals by executing:

```
residuals<-factor.residuals(raqMatrix, pc2$loadings)
```

We can then extract the upper triangle of this matrix using the **upper.tri()** function. This has the effect of extracting only the elements above the diagonal (so we discard the diagonal elements and the elements below the diagonal):

```
residuals<-as.matrix(residuals[upper.tri(residuals)])
```

This command re-creates the object *residuals* by using only the upper triangle of the original matrix. The *as.matrix()* function just makes sure that the residuals are stored as a matrix (they're actually stored as a single column of data). We now have an object called *residuals* that contains the residuals stored in a column. This is handy because it makes it easy to calculate various things. For example, if we want to know how many large residuals there are (i.e., residuals with absolute values greater than 0.05) then we can execute:

```
large.resid<-abs(residuals) > 0.05
```

which uses the *abs()* function to first compute the absolute value of the column of residuals (this is so we ignore whether the residual is positive or negative). The > *0.05* in the command means that *large.resid* will be TRUE (or 1) if the residual is greater than 0.05, and false (or 0) if the residual is less than or equal to 0.05. We end up with a column the same length as the matrix of factor residuals but containing values of TRUE (if the residual is large) or FALSE (if it is small). We can then use the *sum()* function to add up the number of TRUE responses in the matrix:

```
sum(large.resid)
```

[9] **R** has over 3000 packages. For relatively simple things, it's often easier to write a small function yourself than try to find whether a function already exists. Or, you can find a friend that can write a function for you. We will show you how, because we're your friends.

The result is 91. If we want to know this as a proportion of the total number of residuals we can simply execute:

```
sum(large.resid)/nrow(residuals)
```

Executing this command will return the number of large residuals (*sum(large.resid)*) divided by the total number of residuals: *nrows()* tells us how many items (i.e., residuals) there are in total. This will return a value of 0.3596, or 36%. There are no hard and fast rules about what proportion of residuals should be below 0.05; however, if more than 50% are greater than 0.05 you probably have grounds for concern. For our data, we have 36% so we need not worry.

Another way to look at the residuals is to look at their mean. Rather than looking at the mean, we should square the residuals, find the mean, and then find the square root. This is the root-mean-square residual. Again, this is easy to calculate from our *residuals* object. We can execute:

```
sqrt(mean(residuals^2))
```

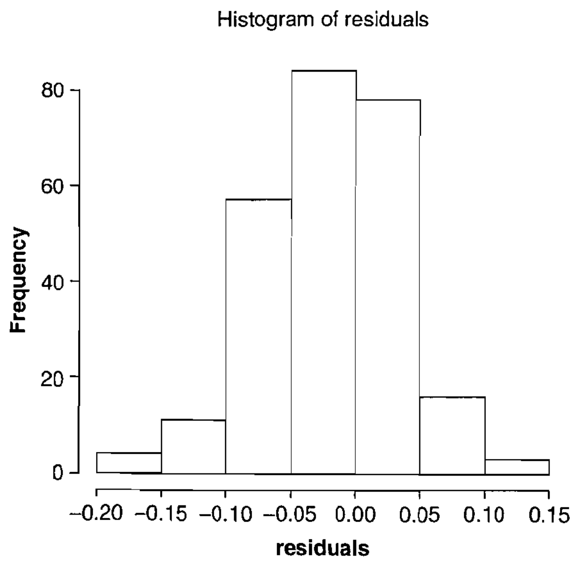
This command squares each item in the residuals object (*residuals* ^ 2), then uses the *mean()* function to compute the mean of these squared residuals. The *sqrt()* function is then used to compute the square root of that mean. The resulting value is 0.055, that's our mean residual. A little lower would have been nice, but this is not dreadful. If this were much higher (say 0.08) we might want to consider extracting more factors.

Finally, it's worth looking at the distributions of the residuals – we expect the residuals to be approximately normally distributed – if there are any serious outliers, even if the other values are all good, we should probably look further into that. We can again use our residuals object to plot a quick histogram using the *hist()* function:

```
hist(residuals)
```

Figure 17.8 shows the histogram of the residuals. They do seem approximately normal and there are no outliers. We could wrap these commands up in a nice function called *residual.stats ()* so that we can use it again in other factor analyses (R's Souls' Tip 17.3).

FIGURE 17.8 Histogram of the model residuals



R's Souls' Tip 17.3 Creating a *residual.stats()* function ②

We saw (in R's Souls' Tip 6.2) that you can write your own functions in **R**. If we wanted to wrap all of the factor analysis residual commands into a function we can do this fairly easily by executing:

```
residual.stats<-function(matrix){
        residuals<-as.matrix(matrix[upper.tri(matrix)])
        large.resid<-abs(residuals) > 0.05
        numberLargeResids<-sum(large.resid)
        propLargeResid<-numberLargeResids/nrow(residuals)
        rmsr<-sqrt(mean(residuals^2))

        cat("Root means squared residual = ", rmsr, "\n")
        cat("Number of absolute residuals > 0.05 = ", numberLargeResids, "\n")
        cat("Proportion of absolute residuals > 0.05 = ", propLargeResid, "\n")
        hist(residuals)
}
```

The first line creates the function by naming it *residual.stats* and telling it to expect a matrix as input. The commands within { } are explained within the main text: they extract the residuals from the matrix entered into the function, compute the number (*numberLargeResids*) and proportion (*propLargeResid*) of absolute values greater than 0.05, compute the root mean squared residual (*rmsr*), and plot a histogram. The commands using the *cat()* function simply specify the text and values to appear in the output.

Having executed the function, we could use it on our residual matrix in one of two ways. First, we could calculate the residual matrix using the *factor.residuals()* function, and label the resulting matrix *resids*. Then pop this matrix into the *residual.stats()* function:

```
resids <- factor.residuals(raqMatrix, pc2$loadings)
residual.stats(resids)
```

The second way is to combine these steps and calculate the residuals matrix directly inside the *residual.stats()* function:

```
residual.stats(factor.residuals(raqMatrix, pc2$loadings))
```

The output would be as follows (and the histogram in Figure 17.8):

```
Root means squared residual =  0.05549286
Number of absolute residuals > 0.05 =  91
Proportion of absolute residuals > 0.05 =  0.3596838
```

CRAMMING SAM'S TIPS Factor extraction

- To decide how many factors to extract, look at the eigenvalues and the scree plot.
- If you have fewer than 30 variables then using eigenvalues greater than 1 is OK (Kaiser's criterion) as long as your communalities are all over .7. Likewise, if your sample size exceeds 250 and the average of the communalities is .6 or greater then this is also fine. Alternatively, with 200 or more participants the scree plot can be used.
- Check the residuals and make sure that fewer than 50% have absolute values greater than 0.05, and that the model fit is greater than 0.90.

17.6.4. Rotation ②

We have already seen that the interpretability of factors can be improved through rotation. Rotation maximizes the loading of each variable on one of the extracted factors while minimizing the loading on all other factors. This process makes it much clearer which variables relate to which factors. Rotation works through changing the absolute values of the variables while keeping their differential values constant. I've discussed the various rotation options in section 17.3.9.1, but, to summarize, the exact choice of rotation will depend on whether or not you think that the underlying factors should be related. If there are theoretical grounds to think that the factors are independent (unrelated) then you should choose one of the orthogonal rotations (I recommend varimax). However, if theory suggests that your factors might correlate then one of the oblique rotations (oblimin or promax) should be selected.

17.6.4.1. Orthogonal rotation (varimax) ②

To carry out a varimax rotation, we change the *rotate* option in the *principal()* function from *"none"* to *"varimax"* (we could also exclude it altogether because varimax is the default if the option is not specified):

```
pc3 <-  principal(raqData, nfactors = 4, rotate = "varimax")
pc3 <-  principal(raqMatrix, nfactors = 4, rotate = "varimax")
```

The first command is to run the analysis from the raw data and the second is if you're using the correlation matrix. In both cases the commands create a model called *pc3* that is the same as the previous model (*pc2*) except that we have used varimax rotation on the model. We can look at this model by executing its name:

```
pc2
```

Output 17.8 shows the first part of the rotated component matrix (also called the rotated factor matrix), which is a matrix of the factor loadings for each variable on each factor. This matrix contains the same information as the component matrix in Output 17.5, except that it is calculated *after* rotation. Notice that the loadings have changed, but the *h2* (communality) and *u2* (uniqueness) columns have not. Rotation changes factors to distribute the variance differently, but it cannot account for more or less variance in the variables than it could before rotation. Also notice that the eigenvalues (SS loadings) have changed. One of the aims of rotation is to even up the eigenvalues; however, the sum of the eigenvalues (and the proportion of variance accounted for) cannot change during rotation.

Interpreting the factor loading matrix is a little complex, and we can make it easier by using the **print.psych()** function. This does two things: first, it removes loadings that are below a certain value that we specify (by using the *cut* option); and second, it reorders the items to try to put them into their factors, which we request using the *sort* option. Generally you should be very careful with the cut-off value – if you think that a loading of .4 will be interesting, you should use a lower cut-off (say, .3), because you don't want to miss a loading that was .39. Execute this command:

```
print.psych(pc3, cut = 0.3, sort = TRUE)
```

This command prints the factor loading matrix associated with the model *pc3*, but displaying only loadings above .3 (*cut* = *0.3*) and sorting items by the size of their loadings (*sort* = *TRUE*).

```
Principal Components Analysis
Call: principal(r = raqData, nfactors = 4, residuals = TRUE, rotate =
"varimax")
```

```
Standardized loadings based upon correlation matrix
      RC3   RC1   RC4   RC2   h2   u2
Q01  0.24  0.50  0.36  0.06 0.43 0.57
Q02 -0.01 -0.34  0.07  0.54 0.41 0.59
Q03 -0.20 -0.57 -0.18  0.37 0.53 0.47
Q04  0.32  0.52  0.31  0.04 0.47 0.53
Q05  0.32  0.43  0.24  0.01 0.34 0.66
Q06  0.80 -0.01  0.10 -0.07 0.65 0.35
Q07  0.64  0.33  0.16 -0.08 0.55 0.45
...
                  RC3  RC1  RC4  RC2
SS loadings      3.73 3.34 2.55 1.95
```

Output 17.8

The resulting matrix is in Output 17.9. Compare this matrix to the unrotated solution (Output 17.5). Before rotation, most variables loaded highly on the first factor and the remaining factors didn't really get a look in. However, the rotation of the factor structure has clarified things considerably: there are four factors and variables load very highly onto only one factor (with the exception of one question). The suppression of loadings less than .3 and ordering variables by loading size also make interpretation considerably easier (because you don't have to scan the matrix to identify substantive loadings).

The next step is to look at the content of questions that load onto the same factor to try to identify common themes. If the mathematical factor produced by the analysis represents some real-world construct then common themes among highly loading questions can help us identify what the construct might be. The questions that load highly on factor 1 are Q6 (I have little experience of computers) with the highest loading of .80, Q18 (R always crashes when I try to use it), Q13 (I worry I will cause irreparable damage ...), Q7 (All computers hate me), Q14 (Computers have minds of their own ...), Q10 (Computers are only for games), and Q15 (Computers are out to get me) with the lowest loading of .46. All these items seem to relate to using computers or R. Therefore we might label this factor *fear of computers*.

Looking at factor 2, we have Q20 (Everybody looks at me when I use R), with a loading of .68, Q21 (I wake up under my duvet ...), Q3 (Standard deviations excite me),[10] Q12 (People try to tell you that R makes statistics easier ...), Q4 (I dream that Pearson is attacking me), Q16 (I weep openly at the mention of central tendency), Q1 (Statistics makes me cry) and Q5 (I don't understand statistics), with the lowest loading of .52 – this item also loads moderately on some of the other factors. The questions that load highly on factor 2 all seem to relate to different aspects of statistics; therefore, we might label this factor *fear of statistics*.

```
Principal Components Analysis
Call: principal(r = raqData, nfactors = 4, rotate = "varimax")
Standardized loadings based upon correlation matrix
    item  RC3   RC1   RC4   RC2   h2   u2
Q06    6  0.80                   0.65 0.35
Q18   18  0.68  0.33             0.60 0.40
Q13   13  0.65                   0.54 0.46
Q07    7  0.64  0.33             0.55 0.45
Q14   14  0.58  0.36             0.49 0.51
Q10   10  0.55                   0.33 0.67
Q15   15  0.46                   0.38 0.62
Q20   20        0.68             0.48 0.52
Q21   21        0.66             0.55 0.45
```

[10] Note that this variable has a negative loading – this means that a high score on the factor is associated with a lower score on this item.

```
Q03    3       -0.57         0.37 0.53 0.47
Q12   12  0.47  0.52              0.51 0.49
Q04    4  0.32  0.52  0.31        0.47 0.53
Q16   16  0.33  0.51  0.31        0.49 0.51
Q01    1        0.50  0.36        0.43 0.57
Q05    5  0.32  0.43              0.34 0.66
Q08    8              0.83        0.74 0.26
Q17   17              0.75        0.68 0.32
Q11   11              0.75        0.69 0.31
Q09    9                     0.65 0.48 0.52
Q22   22                     0.65 0.46 0.54
Q23   23                     0.59 0.41 0.59
Q02    2       -0.34         0.54 0.41 0.59
Q19   19       -0.37         0.43 0.34 0.66

                 RC3  RC1  RC4  RC2
SS loadings     3.73 3.34 2.55 1.95
Proportion Var 0.16 0.15 0.11 0.08
Cumulative Var 0.16 0.31 0.42 0.50

Test of the hypothesis that 4 factors are sufficient.

The degrees of freedom for the null model are 253 and the
objective function was  7.55
The degrees of freedom for the model are 167 and the objective
function was 1.03
The number of observations was 2571 with Chi Square = 2634.37
with prob < 0

Fit based upon off diagonal values = 0.96
```

Output 17.9

Factor 3 has only three items loading on it. Q8 (I have never been good at mathematics), Q17 (I slip into a coma when I see an equation), and Q11 (I did badly at mathematics at school). The three questions that load highly on factor 3 all seem to relate to mathematics; therefore, we might label this factor *fear of mathematics*.

Finally, the questions that load highly on factor 4 are Q9 (My friends are better at statistics than me), Q22 (My friends are better at R), Q2 (My friends will think I'm stupid) and Q19 (Everybody looks at me). All these items contain some component of social evaluation from friends; therefore, we might label this factor *peer evaluation*.

This analysis seems to reveal that the initial questionnaire, in reality, is composed of four subscales: fear of computers, fear of statistics, fear of maths and fear of negative peer evaluation. There are two possibilities here. The first is that the RAQ failed to measure what it set out to (namely, R anxiety) but does measure some related constructs. The second is that these four constructs are sub-components of R anxiety; however, the factor analysis does not indicate which of these possibilities is true.

17.6.4.2. Oblique rotation ②

When we did the orthogonal rotation, we told R that we expected the components that we extracted to be uncorrelated. This was a bit of a strange thing to say. All of our factors related to fear: fear of computers, fear of statistics, fear of negative peer evaluation and feed of mathematics. It's likely that these will be correlated: people with fear of one thing might have fear of other things. If this is the case an oblique rotation is called for.

The command for an oblique rotation is very similar to that for an orthogonal rotation, we just change the *rotate* option, from *"varimax"* to *"oblimin"*:

```
pc4 <- principal(raqData, nfactors = 4, rotate = "oblimin")
pc4 <- principal(raqMatrix, nfactors = 4, rotate = "oblimin")
```

The first command is to run the analysis from the raw data and the second is if you're using the correlation matrix. In both cases the commands create a model called *pc4* that is the same as the model *pc2* except that we have used oblimin rotation on the model. As with the previous model, we can look at the factor loadings from this model in a nice easy-to-digest format by executing:

```
print.psych(pc4, cut = 0.3, sort = TRUE)
```

The output from this analysis is shown in Output 17.10. The same four factors seem to have emerged although they are in a different order. Factor 1 seems to represent fear of computers, factor 2 represents fear of peer evaluation, factor 3 represents fear of statistics and factor 4 represents fear of mathematics.

```
Principal Components Analysis
Call: principal(r = raqData, nfactors = 4, rotate = "oblimin")
Standardized loadings based upon correlation matrix
     item  TC1   TC4   TC3   TC2   h2   u2
Q06     6  0.87                   0.65 0.35
Q18    18  0.70                   0.60 0.40
Q07     7  0.64                   0.55 0.45
Q13    13  0.64                   0.54 0.46
Q10    10  0.57                   0.33 0.67
Q14    14  0.57                   0.49 0.51
Q12    12  0.45        0.43       0.51 0.49
Q15    15  0.40                   0.38 0.62
Q08     8        0.90             0.74 0.26
Q11    11        0.78             0.69 0.31
Q17    17        0.78             0.68 0.32
Q20    20              0.71       0.48 0.52
Q21    21              0.60       0.55 0.45
Q03     3             -0.51       0.53 0.47
Q04     4              0.41       0.47 0.53
Q16    16              0.41       0.49 0.51
Q01     1        0.33  0.40       0.43 0.57
Q05     5              0.34       0.34 0.66
Q22    22                    0.65 0.46 0.54
Q09     9                    0.63 0.48 0.52
Q23    23                    0.61 0.41 0.59
Q02     2             -0.36  0.51 0.41 0.59
Q19    19             -0.35  0.38 0.34 0.66

                TC1  TC4  TC3  TC2
SS loadings     3.90 2.88 2.94 1.85
Proportion Var 0.17 0.13 0.13 0.08
Cumulative Var 0.17 0.29 0.42 0.50

 With factor correlations of
      TC1   TC4   TC3   TC2
TC1  1.00  0.44  0.36 -0.18
TC4  0.44  1.00  0.31 -0.10
TC3  0.36  0.31  1.00 -0.17
TC2 -0.18 -0.10 -0.17  1.00
```

```
Test of the hypothesis that 4 factors are sufficient.
The degrees of freedom for the null model are  253 and the
objective function was 7.55
The degrees of freedom for the model are 167 and the objective
function was 1.03
The number of observations was 2571 with Chi Square = 2634.37
with prob < 0

Fit based upon off diagonal values = 0.96
```

Output 17.10

Also in this output you'll find a correlation matrix between the factors. This matrix contains the correlation coefficients between factors – **R** didn't bother to show this to us when it did an orthogonal rotation, because the correlations were all zero. Factor 2 (*TC2*) has little relationship with any other factors (the correlation coefficients are low), but all other factors are interrelated to some degree (notably *TC3* with both *TC1* and *TC4*, and *TC4* with *TC1*). The fact that these correlations exist tell us that the constructs measured can be interrelated. If the constructs were independent then we would expect oblique rotation to provide an identical solution to an orthogonal rotation and the component correlation matrix should be an identity matrix (i.e., all factors have correlation coefficients of 0). Therefore, this final matrix gives us a guide to whether it is reasonable to assume independence between factors: for these data it appears that we cannot assume independence. Therefore, the results of the orthogonal rotation should not be trusted: the obliquely rotated solution is probably more meaningful.

When an oblique rotation is conducted the factor matrix is split into two matrices: the *pattern matrix* and the *structure matrix* (see Jane Superbrain Box 17.1). For orthogonal rotation these matrices are the same. The pattern matrix contains the factor loadings and is comparable to the factor matrix that we interpreted for the orthogonal rotation. The structure matrix takes into account the relationship between factors (in fact it is a product of the pattern matrix and the matrix containing the correlation coefficients between factors). Most researchers interpret the pattern matrix, because it is usually simpler; however, there are situations in which values in the pattern matrix are suppressed because of relationships between the factors. Therefore, the structure matrix is a useful double-check and Graham et al. (2003) recommend reporting both (with some useful examples of why this can be important).

Getting the structure matrix out of R is a little bit more complex than getting the pattern matrix. You need to multiply the factor loading matrix by the correlation matrix of the factors. We've come across the loadings, these are called *pc4$loadings*. The correlations of the factors are called the Phi (Greek letter ϕ, which rhymes with pie) and so are stored in *pc4$Phi*. Given that we have these two matrices, we can get the structure matrix by multiplying them; however, this is not a regular multiplication, this is a matrix multiplication, so instead of writing * we write %*%. The structure matrix is therefore given by executing:

```
pc4$loadings %*% pc4$Phi
```

The kind of people that write **R** think that this is straightforward, but we realize it's not, especially when you're starting out. Also, doing this calculation produces a rather unfriendly looking structure matrix that isn't sorted by the size of factor loadings. So, we've written a function for you, called **factor.structure()**; you can source it from our DSUR package. The function takes this general form:

```
factor.structure(pcModel, cut = 0.2, decimals = 2)
```

All you need to do is enter the name of the principal components model into the function and execute. Just like the *print.psych()* function we have included an option (*cut*) so you can specify a value below which you don't want to see the loading (the default is .2), and also an option, *decimals*, that allows you to change the number of decimal places you see (the default is 2). For our current model we could execute:

```
factor.structure(pc4, cut = 0.3)
```

Output 17.11 shows the structure matrix. The picture becomes more complicated in the structure matrix because with the exception of factor 2, several variables load quite highly onto more than one factor. This has occurred because of the relationship between factors 1 and 3 and between factors 3 and 4. This example should highlight why the pattern matrix is preferable for interpretative reasons: because it contains information about the *unique* contribution of a variable to a factor.

```
       TC1    TC4    TC3   TC2
Q06   0.78
Q18   0.76   0.36   0.42
Q13   0.72   0.43   0.33
Q07   0.72   0.38   0.42
Q14   0.67   0.35   0.44
Q12    0.6   0.33   0.59
Q10   0.56
Q15   0.55   0.44   0.31
Q08          0.85
Q17   0.44   0.82    0.3
Q11   0.43   0.82
Q21   0.46   0.37    0.7
Q20                 0.68
Q03  -0.39  -0.36  -0.64  0.41
Q16    0.5    0.5   0.58
Q04   0.47   0.49   0.56
Q01    0.4    0.5   0.53
Q05   0.44    0.4   0.47
Q22                       0.66
Q09                       0.66
Q23                       0.58
Q02                -0.39  0.55
Q19                -0.44  0.45
```

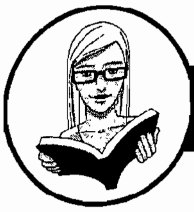
Output 17.11

On a theoretical level the dependence between our factors does not cause concern; we might expect a fairly strong relationship between fear of maths, fear of statistics and fear of computers. Generally, the less mathematically and technically minded people struggle with statistics. However, we would not expect these constructs to correlate with fear of peer evaluation (because this construct is more socially based). In fact, this factor is the one that correlates fairly badly with all others – so on a theoretical level, things have turned out rather well!

17.6.5. Factor scores ②

Having reached a suitable solution and rotated that solution, we can look at the factor scores. Factor scores are obtained by adding *scores = TRUE* to the *principal()* function. Therefore, to get factor scores for our model *pc4*, we would rerun the analysis using by executing:

```
pc5 <- principal(raqData, nfactors = 4, rotate = "oblimin", scores = TRUE)
```



CRAMMING SAM'S TIPS Interpretation

- If you've conducted orthogonal rotation then look at the table labelled *rotated component matrix*. For each variable, note the component for which the variable has the highest loading. Also, for each component, note the variables that load highly onto it (by 'high' I mean loadings should be above .4 when you ignore the plus or minus sign). Try to make sense of what the factors represent by looking for common themes in the items that load onto them.
- If you've conducted oblique rotation then calculate and look at the *pattern matrix*. For each variable, note the component for which the variable has the highest loading. Also, for each component, note the variables that load highly onto it (by 'high' I mean loadings should be above .4 when you ignore the plus or minus sign). Double-check what you find by doing the same thing for the *structure matrix*. Try to make sense of what the factors represents by looking for common themes in the items that load onto them.

By setting the *scores* option to *TRUE* the factor scores are added to the principal component model in an object called *scores*; therefore, we can access these scores by using *pc5$scores* (which translates as the *scores* object attached to the model *pc5* that we just created). To view the factor scores, you could execute:

```
pc5$scores
```

However, there are rather a lot of them (2571 actually), so let's look at the first 10 rows, by using the *head()* function and executing:

```
head(pc5$scores, 10)
```

SELF-TEST

✓ Using what you learnt in Chapter 6, or Section 17.6.2, calculate the correlation matrix for the factor scores. Compare this to the correlations of the factors in Output 17.10.

```
              TC1        TC4         TC3        TC2
 [1,]  0.37296709  1.8808424  0.95979596  0.3910711
 [2,]  0.63334164  0.2374679  0.29090777 -0.3504080
 [3,]  0.39712768 -0.1056263 -0.09333769  0.9249353
 [4,] -0.78741595  0.2956628 -0.77703307  0.2605666
 [5,]  0.04425942  0.6815179  0.59786611 -0.6912687
 [6,] -1.70018648  0.2091685  0.02784164  0.6653081
 [7,]  0.66139239  0.4224096  1.52552021 -0.9805434
 [8,]  0.59491329  0.4060248  1.06465956 -1.0932598
 [9,] -2.34971189 -3.6134797 -1.42999472 -0.5443773
[10,]  0.93504597  0.2285419  0.96735727 -1.5712753
```

Output 17.12

Output 17.12 shows the factor scores for the first 10 participants. Factor scores can be used in this way to assess the relative fear of one person compared to another. We can also use factor scores in regression when groups of predictors correlate so highly that there is multicollinearity.

Before we can do any analysis with our factor scores, we need to add the factor scores into our dataframe. To do this, we use the *cbind()* function, which we have used numerous times before:

```
raqData <- cbind(raqData, pc5$scores)
```

SELF-TEST

✓ Can you think of another way of obtaining the structure matrix (the correlations between factors and items) now you've learned about factor scores?

17.6.6. Summary ②

To sum up, the analyses revealed four underlying scales in our questionnaire that may, or may not, relate to genuine sub-components of R anxiety. It also seems as though an obliquely rotated solution was preferred due to the interrelationships between factors. The use of factor analysis is purely exploratory; it should be used only to guide future hypotheses, or to inform researchers about patterns within data sets. A great many decisions are left to the researcher using factor analysis, and I urge you to make informed decisions, rather than basing decisions on the outcomes you would like to get. In section 17.9 we consider whether or not our scale is reliable.

17.7. How to report factor analysis ①

As with any analysis, when reporting factor analysis we need to provide our readers with enough information to make an informed opinion about our data. As a bare minimum we should be very clear about our criteria for extracting factors and the method of rotation used. We must also produce a table of the rotated factor loadings of all items and flag (in bold) values above a criterion level (I would personally choose .40, but I discussed the various criteria you could use in section 17.3.9.2). You should also report the percentage of variance that each factor explains and possibly the eigenvalue too. Table 17.1 shows an example of such a table for the RAQ data; note that I have also reported the sample size in the title.

In my opinion, a table of factor loadings and a description of the analysis are a bare minimum, though. You could consider (if it's not too large) including the table of correlations from which someone could reproduce your analysis (should they want to). You could also consider including some information on sample size adequacy.

For this example we might write something like this:

✓ A principal components analysis (PCA) was conducted on the 23 items with orthogonal rotation (varimax). The Kaiser–Meyer–Olkin measure verified the sampling adequacy for the analysis KMO = .93 ('superb' according to Kaiser, 1974), and all KMO values for individual items were > .77, which is well above the acceptable limit of .5. Bartlett's test of sphericity, χ^2 (253) = 19,334, p < .001, indicated that correlations between items were sufficiently large for PCA. An initial analysis was run to obtain eigenvalues for each component in the data. Four components had eigenvalues over Kaiser's criterion of 1 and in combination explained 50.32% of the variance. The scree plot was slightly ambiguous and showed inflexions that would justify retaining both two and four components. Given the large sample size, and the convergence of the scree plot and Kaiser's criterion on four components, four components were retained in the final analysis. Table 17.1 shows the factor loadings after rotation. The items that cluster on the same components suggest that component 1 represents a fear of computers, component 2 a fear of statistics, component 3 a fear of maths and component 4 peer evaluation concerns.

Table 17.1 Summary of exploratory factor analysis results for the **R** anxiety questionnaire ($N = 2571$)

	Varimax rotated factor loadings			
Item	*Fear of computers*	*Fear of statistics*	*Peer evaluation*	*Fear of maths*
I have little experience of computers	**.80**	−.01	−.07	.10
R always crashes when I try to use it	**.68**	.33	−.08	.13
I worry that I will cause irreparable damage because of my incompetence with computers	**.65**	.23	−.10	.23
All computers hate me	**.64**	.33	−.08	.16
Computers have minds of their own and deliberately go wrong whenever I use them	**.58**	.36	−.07	.14
Computers are useful only for playing games	**.55**	.00	−.12	.13
Computers are out to get me	**.46**	.22	−.19	.29
I can't sleep for thoughts of eigen vectors	−.04	**.68**	−.14	.08
I wake up under my duvet thinking that I am trapped under a normal distribution	.29	**.66**	−.07	.16
Standard deviations excite me	−.20	**−.57**	.37	−.18
People try to tell you that R makes statistics easier to understand but it doesn't	**.47**	**.52**	−.08	.10
I dream that Pearson is attacking me with correlation coefficients	.32	**.52**	.04	.31
I weep openly at the mention of central tendency	.33	**.51**	−.12	.31
Statistics makes me cry	.24	**.50**	.06	.36
I don't understand statistics	.32	**.43**	.02	.24
I have never been good at mathematics	.13	.17	.01	**.83**
I slip into a coma whenever I see an equation	.27	.22	−.04	**.75**
I did badly at mathematics at school	.26	.21	−.14	**.75**
My friends are better at statistics than me	−.09	−.20	**.65**	.12
My friends are better at R than I am	−.19	.03	**.65**	−.10
If I'm good at statistics my friends will think I'm a nerd	−.02	.17	**.59**	−.20
My friends will think I'm stupid for not being able to cope with R	−.01	−.34	**.54**	.07
Everybody looks at me when I use R	−.15	−.37	**.43**	−.03
Eigenvalues	3.73	3.34	1.95	2.55
% of variance	16.22	14.52	8.48	11.10
α	.82	.82	.57	.82

Note: Factor loadings over .40 appear in bold.

Finally, if you have used oblique rotation you should consider reporting a table of both the structure and pattern matrix because the loadings in these tables have different interpretations (see Jane Superbrain Box 17.1).

Labcoat Leni's Real Research 17.1 World wide addiction? ②

Nichols, L. A., & Nicki, R. (2004). *Psychology of Addictive Behaviors*, *18*(4), 381–384.

The Internet is now a houshold tool. In 2007 it was estimated that around 179 million people worldwide used the Internet (over 100 million of those were in the USA and Canada). From the increasing popularity (and usefulness) of the Internet has emerged a new phenomenon: Internet addiction. This is now a serious and recognized problem, but until very recently it was very difficult to research this topic because there was not a psychometrically sound measure of Internet addition. That is, until Laura Nichols and Richard Nicki developed the Internet Addiction Scale, IAS (Nichols & Nicki, 2004). (Incidentally, while doing some research on this topic I encountered an Internet addiction recovery website that I won't name but that offered a whole host of resources that would keep you online for ages, such as questionnaires, an online support group, videos, articles, a recovery blog and podcasts. It struck me that this was a bit like having a recovery centre for heroin addiction where the addict arrives to be greeted by a nice-looking counsellor who says 'there's a huge pile of heroin in the corner over there, just help yourself'.)

Anyway, Nichols and Nicki developed a 36-item questionnaire to measure internet addiction. It contained items such as 'I have stayed on the Internet longer than I intended to' and 'My grades/work have suffered because of my Internet use', which could be responded to on a 5-point scale (Never, Rarely, Sometimes, Frequently, Always). They collected data from 207 people to validate this measure.

The data from this study are in the file **Nichols & Nicki (2004).dat**. The authors dropped two items because they had low means and variances, and dropped three others because of relatively low correlations with other items. They performed a principal components analysis on the remaining 31 items. Labcoat Leni wants you to run some descriptive statistics to work out which two items were dropped for having low means/variances, then inspect a correlation matrix to find the three items that were dropped for having low correlations. Finally, he wants you to run a principal components analysis on the data.

Answers are in the additional material on the companion website (or look at the original article).

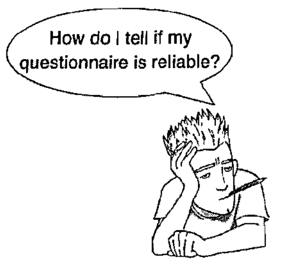
17.8. Reliability analysis ②

17.8.1. Measures of reliability ③

If you're using factor analysis to validate a questionnaire, it is useful to check the reliability of your scale.

SELF-TEST

✓ Thinking back to Chapter 1, what are reliability and test–retest reliability?



Reliability means that a measure (or in this case questionnaire) should consistently reflect the construct that it is measuring. One way to think of this is that, other things being equal, a person should get the same score on a questionnaire if they complete it at two different points in time (we have already discovered that this is called test–retest reliability). So, someone who is terrified of statistics and who scores highly on our RAQ should score similarly highly if we tested them a month later (assuming they hadn't gone into some kind of statistics-anxiety therapy in that month). Another way to look at reliability is to say that two people who are the same in terms of the construct being measured should get the same score. So, if we took two people who were equally statistics-phobic, then they should get more or less identical scores on the RAQ. Likewise, if we took two people who loved statistics, they should both get equally low scores. It should be apparent that if we took someone who loved statistics and someone who was terrified of it, and they got the same score on our questionnaire, then it wouldn't be an accurate measure of statistical anxiety. In statistical terms, the usual way to look at reliability is based on the idea that individual items (or sets of items) should produce results consistent with the overall questionnaire. So, if we take someone scared of statistics, then their overall score on the RAQ will be high; if the RAQ is reliable then if we randomly select some items from it the person's score on those items should also be high.

The simplest way to do this in practice is to use **split-half reliability**. This method randomly splits the data set into two. A score for each participant is then calculated based on each half of the scale. If a scale is very reliable a person's score on one half of the scale should be the same (or similar) to their score on the other half: therefore, across several participants, scores from the two halves of the questionnaire should correlate perfectly (well, very highly). The correlation between the two halves is the statistic computed in the split-half method, with large correlations being a sign of reliability. The problem with this method is that there are several ways in which a set of data can be split into two and so the results could be a product of the way in which the data were split. To overcome this problem, Cronbach (1951) came up with a measure that is loosely equivalent to splitting data in two in every possible way and computing the correlation coefficient for each split. The average of these values is equivalent to **Cronbach's alpha**, α, which is the most common measure of scale reliability.[11]

Cronbach's α is:

$$\alpha = \frac{N^2\overline{Cov}}{\sum s^2_{item} + \sum Cov_{item}} \tag{17.6}$$

which may look complicated, but actually isn't. The first thing to note is that for each item on our scale we can calculate two things: the variance within the item, and the covariance between a particular item and any other item on the scale. Put another way, we can construct a variance–covariance matrix of all items. In this matrix the diagonal elements will be the variance within a particular item, and the off-diagonal elements will be covariances between pairs of items. The top half of the equation is simply the number of items (N) squared multiplied by the average covariance between items (the average of the off-diagonal elements in the aforementioned variance–covariance matrix). The bottom half is

[11] Although this is the easiest way to conceptualize Cronbach's α, whether or not it is exactly equal to the average of all possible split-half reliabilities depends on exactly how you calculate the split-half reliability (see the glossary for computational details). If you use the Spearman–Brown formula, which takes no account of item standard deviations, then Cronbach's α will be equal to the average split-half reliability only when the item standard deviations are equal; otherwise α will be smaller than the average. However, if you use a formula for split-half reliability that does account for item standard deviations (such as Flanagan, 1937; Rulon, 1939) then α will always equal the average split-half reliability (see Cortina, 1993).

just the sum of all the item variances and item covariances (i.e., the sum of everything in the variance–covariance matrix).

There is a standardized version of the coefficient too, which essentially uses the same equation except that correlations are used rather than covariances, and the bottom half of the equation uses the sum of the elements in the correlation matrix of items (including the ones that appear on the diagonal of that matrix). The normal alpha is appropriate when items on a scale are summed to produce a single score for that scale (the standardized alpha is not appropriate in these cases). The standardized alpha is useful, though, when items on a scale are standardized before being summed.

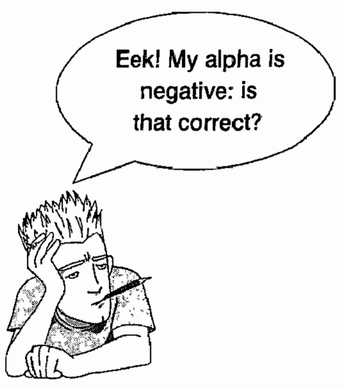
17.8.2. Interpreting Cronbach's α (some cautionary tales ...) ②

You'll often see in books, journal articles, or be told by people that a value of .7 to .8 is an acceptable value for Cronbach's α; values substantially lower indicate an unreliable scale. Kline (1999) notes that although the generally accepted value of .8 is appropriate for cognitive tests such as intelligence tests, for ability tests a cut-off point of .7 is more suitable. He goes on to say that when dealing with psychological constructs values below even .7 can, realistically, be expected because of the diversity of the constructs being measured.

However, Cortina (1993) notes that such general guidelines need to be used with caution because the value of α depends on the number of items on the scale. You'll notice that the top half of the equation for α includes the number of items squared. Therefore, as the number of items on the scale increases, α will increase. Therefore, it's possible to get a large value of α because you have a lot of items on the scale! For example, Cortina reports data from two scales, both of which have $\alpha = .8$. The first scale has only three items, and the average correlation between items was a respectable .57; however, the second scale had 10 items, with an average correlation between these items of a less respectable .28. Clearly the internal consistency of these scales differs enormously, yet they are both equally reliable

A second common interpretation of alpha is that it measures 'unidimensionality', or the extent to which the scale measures one underlying factor or construct. This interpretation stems from the fact that when there is one factor underlying the data, α is a measure of the strength of that factor (see Cortina, 1993). However, Grayson (2004) demonstrates that data sets with the same α can have very different structures. He showed that $\alpha = .8$ can be achieved in a scale with one underlying factor, with two moderately correlated factors and with two uncorrelated factors. Cortina (1993) has also shown that with more than 12 items, and fairly high correlations between items ($r > .5$), α can reach values around and above .7 (.65 to .84). These results compellingly show that α should not be used as a measure of 'unidimensionality'. Indeed, Cronbach (1951) suggested that if several factors exist then the formula should be applied separately to items relating to different factors. In other words, if your questionnaire has subscales, α should be applied separately to these subscales.

The final warning is about items that have a reverse phrasing. For example, in our RAQ that we used in the factor analysis part of this chapter, we had one item (question 3) that was phrased the opposite way around to all other items. The item was 'standard deviations excite me'. Compare this to any other item and you'll see it requires the opposite response. For example, item 1 is 'statistics make me cry'. Now, if you don't like statistics then you'll strongly agree with this statement and so will get a score of 5 on our scale. For item 3, if you hate statistics then standard deviations are unlikely to excite you so you'll strongly disagree and get a score of 1 on the scale. These reverse-phrased items are important for reducing response bias; participants will actually have to read the items in case they are phrased the other way around. For factor analysis, this reverse phrasing doesn't matter, all that happens is you get a negative factor loading for any reversed items (in fact, look



at Output 17.10 and you'll see that item 3 has a negative factor loading). However, in reliability analysis these reverse-scored items do make a difference. To see why, think about the equation for Cronbach's α. In this equation, the top half incorporates the *average* covariance between items. If an item is reverse-phrased then it will have a negative relationship with other items, hence the covariances between this item and other items will be negative. The average covariance is obviously the sum of covariances divided by the number of covariances, and by including a bunch of negative values we reduce the sum of covariances, and hence we also reduce Cronbach's α, because the top half of the equation gets smaller. In extreme cases, it is even possible to get a negative value for Cronbach's α, simply because the magnitude of negative covariances is bigger than the magnitude of positive ones. A negative Cronbach's α doesn't make much sense, but it does happen, and if it does, ask yourself whether you included any reverse-phrased items.

17.8.3. Reliability analysis with R Commander ①

As with factor analysis, it's possible to use R Commander to obtain reliability estimates. However, the procedure is not as flexible as the *alpha()* function in the psych package, so that's the one we use.

17.8.4. Reliability analysis using R ②

Let's test the reliability of the RAQ using the data in RAQ.dat. Remember also that I said we should conduct reliability analysis on any subscales individually. If we use the results from our orthogonal rotation (look back at), then we have four subscales:

1 Subscale 1 (*Fear of computers*): items 6, 7, 10, 13, 14, 15, 18
2 Subscale 2 (*Fear of statistics*): items 1, 3, 4, 5, 12, 16, 20, 21
3 Subscale 3 (*Fear of mathematics*): items 8, 11, 17
4 Subscale 4 (*Peer evaluation*): items 2, 9, 19, 22, 23

(Don't forget that question 3 has a negative sign; we'll need to remember to deal with that.) First, we'll create four new data sets, containing the subscales for the items. We don't need to do that, but it saves a lot of typing later on. We can create these data sets by simply selecting the appropriate columns of the full dataframe (*raqData*) as described in section 3.9.1.

```
computerFear<-raqData[, c(6, 7, 10, 13, 14, 15, 18)]
statisticsFear <- raqData[, c(1, 3, 4, 5, 12, 16, 20, 21)]
mathFear <- raqData[, c(8, 11, 17)]
peerEvaluation <- raqData[, c(2, 9, 19, 22, 23)]
```

This command takes the *raqData* dataframe and retains all of the rows (hence no command before the comma), and any columns specified in the *c()* function after the comma. For example, the first command creates an object called *computerFear* that contains only columns 6, 7, 10, 13, 14, 15, and 18 of the dataframe *raqData*.

Reliability analysis is done with the **alpha()** function, which is found in the *psych* package. You might have a problem here, because there is also a function in *ggplot2* called alpha, and if you've loaded *ggplot2* first, that version will have priority. This was covered in R's Souls' Tip 3.4, but to remind you, if you get the wrong *alpha()* function, you can specify the package, using:

```
psych::alpha()
```

An additional complication that we need to deal with is that pesky item 3, which is negatively scored. We can do one of two things with this item. We can reverse the variable in the data set, or we can tell *alpha()* that it is negative, using the *keys* option. This latter option is better because we leave the initial data unchanged (which is useful because we don't get into awkward situations in which we save the data and then can't recall at a later date whether or not the data contains the reverse scored or original scores).

To use the keys option we give *alpha()* a vector of 1s and −1s, which matches the number of variables in the data set, using a 1 for a positively score item and a −1 for a negatively scored item. So for *computerFear*, which has only positively scored items, we would use:

```
keys = c(1, 1, 1, 1, 1, 1, 1)
```

but for *statisticsFear*, which has item 3 (the negatively scored item) as its second item, we would use:

```
keys = c(1, -1, 1, 1, 1, 1, 1, 1)
```

For three of our four subscales we don't need to use the *keys* option because all items are positively scored, but for *statisticsFear* we need to. To use the *alpha()* function we simply input the name of the dataframe for each subscale, and, where necessary, include the *keys* option. Therefore, we could run the reliability analysis for our four subscales by executing:

```
alpha(computerFear)
alpha(statisticsFear, keys = c(1, -1, 1, 1, 1, 1, 1, 1))
alpha(mathFear)
alpha(peerEvaluation)
```

17.8.5. Interpreting the output ②

Output 17.13 shows the results of this basic reliability analysis for the fear of computing subscale. First, and perhaps most important, the value of *alpha* at the very top is Cronbach's α: the overall reliability of the scale (you should look at the raw alpha, they're usually very similar though). To reiterate, we're looking for values in the range of .7 to .8 (or thereabouts) bearing in mind what we've already noted about effects from the number of items. In this case α is slightly above .8, and is certainly in the region indicated by Kline (1999), so this probably indicates good reliability.

Along with alpha, there is a measure labelled G6, short for Guttman's lambda 6; this can be calculated from the squared multiple correlation (hence it's labelled *smc*).[12] The *average_r* is the average inter-item correlation (from which we can calculate standardized alpha).

Also in this top section are some scale characteristics. If we calculated someone's score by taking the average of all of their items (which is the same as adding up the score and dividing by the number of items), we would have a variable with an overall mean of 3.4 and standard deviation of 0.71.[13]

[12] Fact fiends might be interested to know that Guttman came up with Cronbach's alpha before Cronbach, and called it lambda 3.

[13] You can test this by running:

```
describe(apply((raq[c(6, 7, 10, 13, 14, 15, 18)]), 1, mean))
```

which gives you a mean of 3.42 and *sd* = 0.71.

Next, we get a table giving the statistics for the scale if we deleted each item in turn. The values in the column labelled *raw_alpha* are the values of the overall α if that item isn't included in the calculation. As such, they reflect the change in Cronbach's α that would be seen if a particular item were deleted. The overall α is .82, and so all values in this column should be around that same value. What we're actually looking for is values of alpha greater than the overall α. If you think about it, if the deletion of an item increases Cronbach's α then this means that the deletion of that item improves reliability (remembering that scales with more items are more reliable, so removing an item should always lower alpha). Therefore, any items that have values of α in this column greater than the overall α may need to be deleted from the scale to improve its reliability. None of the items here would substantially affect reliability if they were deleted. None of the items increase alpha by being deleted. This table also contains the standardized alpha if the item is removed, the G6 if the item is removed and the mean correlation if the item is removed.

The next table in the output is labelled *item statistics*. The values in the first column labelled *r* are the correlations between each item and the total score from the questionnaire – sometimes called item–total correlations. There's a problem with this statistic, and that is that the item is included in the total. That is, if we correlate item 6 with the mean of all items, we're correlating the item with itself, so of course it will correlate. We can correct this by correlating each item with all of the other items. Two versions of this are presented, *r.cor* and *r.drop*: *r.cor* is a little complex, so we won't go into it (but the help file for alpha explains it), *r.drop* is the correlation of that item with the scale total if that item isn't included in the scale total. Sometimes this is called the item–rest correlation (because it's how the item correlates with the rest of the items) and sometimes it's called the corrected item–total correlation.

```
Reliability analysis
Call: alpha(x = computerFear)

  raw_alpha std.alpha G6(smc) average_r mean   sd
      0.82      0.82    0.81       0.4  3.4 0.71

 Reliability if an item is dropped:
    raw_alpha std.alpha G6(smc) average_r
Q06      0.79      0.79    0.77      0.38
Q07      0.79      0.79    0.77      0.38
Q10      0.82      0.82    0.80      0.44
Q13      0.79      0.79    0.77      0.39
Q14      0.80      0.80    0.77      0.39
Q15      0.81      0.81    0.79      0.41
Q18      0.79      0.78    0.76      0.38

 Item statistics
       n    r r.cor r.drop mean   sd
Q06 2571 0.74  0.68   0.62  3.8 1.12
Q07 2571 0.73  0.68   0.62  3.1 1.10
Q10 2571 0.57  0.44   0.40  3.7 0.88
Q13 2571 0.73  0.67   0.61  3.6 0.95
Q14 2571 0.70  0.64   0.58  3.1 1.00
Q15 2571 0.64  0.54   0.49  3.2 1.01
Q18 2571 0.76  0.72   0.65  3.4 1.05

Non missing response frequency for each item
       1    2    3    4    5 miss
Q06 0.06 0.10 0.13 0.44 0.27    0
Q07 0.09 0.24 0.26 0.34 0.07    0
```

```
Q10 0.02 0.10 0.18 0.57 0.14    0
Q13 0.03 0.12 0.25 0.48 0.12    0
Q14 0.07 0.18 0.38 0.31 0.06    0
Q15 0.06 0.18 0.30 0.39 0.07    0
Q18 0.06 0.12 0.31 0.37 0.14    0
```

Output 17.13

In a reliable scale all items should correlate with the total. So, we're looking for items that don't correlate with the overall score from the scale: if any of these values of *r.drop* are less than about .3 then we've got problems, because it means that a particular item does not correlate very well with the scale overall. Items with low correlations may have to be dropped. For these data, all data have corrected item–total correlations above .3, which is encouraging. The table also shows the mean and standard deviation of the scale if the item is omitted.

The final table in the alpha output is a table of frequencies. It tells us what percentage of people gave each response to each of the items. This is useful to make sure that everyone in your sample is not giving the same response. It is usually the case that an item where everyone (or almost everyone) gives the same response will almost certainly have poor reliability statistics.

As a final point, it's worth noting that if items do need to be removed at this stage then you should rerun your factor analysis as well to make sure that the deletion of the item has not affected the factor structure.

```
Reliability analysis
Call: alpha(x = statisticsFear, keys = c(1, -1, 1, 1, 1, 1, 1, 1))

  raw_alpha std.alpha G6(smc) average_r mean  sd
      0.82      0.82    0.81      0.37  3.1 0.5

 Reliability if an item is dropped:
    raw_alpha std.alpha G6(smc) average_r
Q01      0.80      0.80    0.79      0.37
Q03      0.80      0.80    0.79      0.37
Q04      0.80      0.80    0.78      0.36
Q05      0.81      0.81    0.80      0.38
Q12      0.80      0.80    0.79      0.36
Q16      0.79      0.80    0.78      0.36
Q20      0.82      0.82    0.80      0.40
Q21      0.79      0.80    0.78      0.36

 Item statistics
       n    r r.cor r.drop mean   sd
Q01 2571 0.67  0.60   0.54  3.6 0.83
Q03 2571 0.67  0.60   0.55  3.4 1.08
Q04 2571 0.70  0.64   0.58  3.2 0.95
Q05 2571 0.63  0.55   0.49  3.3 0.96
Q12 2571 0.69  0.63   0.57  2.8 0.92
Q16 2571 0.71  0.67   0.60  3.1 0.92
Q20 2571 0.56  0.47   0.42  2.4 1.04
Q21 2571 0.71  0.67   0.61  2.8 0.98

Non missing response frequency for each item
       1    2    3    4    5 miss
Q01 0.02 0.07 0.29 0.52 0.11    0
Q03 0.03 0.17 0.34 0.26 0.19    0
Q04 0.05 0.17 0.36 0.37 0.05    0
```

```
Q05 0.04 0.18 0.29 0.43 0.06    0
Q12 0.09 0.23 0.46 0.20 0.02    0
Q16 0.06 0.16 0.42 0.33 0.04    0
Q20 0.22 0.37 0.25 0.15 0.02    0
Q21 0.09 0.29 0.34 0.26 0.02    0
```

Output 17.14

OK, let's move on to the fear of statistics subscale (items 1, 3, 4, 5, 12, 16, 20 and 21). I won't go through the R output in detail again, but it is shown in Output 17.14. The overall α is .82, and none of the items here would increase the reliability if they were deleted. The values in the column labelled *r.drop* are again all above .3, which is good. In all, this indicates that all items are positively contributing to the overall reliability. The overall α is also excellent (.82) because it is above .8, and indicates good reliability.

```
Reliability analysis
Call: alpha(x = statisticsFear)

  raw_alpha std.alpha G6(smc) average_r mean  sd
      0.61      0.64    0.71      0.18  3.1 0.5

 Reliability if an item is dropped:
    raw_alpha std.alpha G6(smc) average_r
Q01      0.52      0.56    0.64      0.15
Q03      0.80      0.80    0.79      0.37
Q04      0.50      0.55    0.64      0.15
Q05      0.52      0.57    0.66      0.16
Q12      0.52      0.56    0.65      0.15
Q16      0.51      0.55    0.63      0.15
Q20      0.56      0.60    0.68      0.18
Q21      0.50      0.55    0.63      0.15

 Item statistics
       n     r r.cor r.drop mean   sd
Q01 2571  0.68  0.62   0.51  3.6 0.83
Q03 2571 -0.37 -0.64  -0.55  3.4 1.08
Q04 2571  0.69  0.65   0.53  3.2 0.95
Q05 2571  0.65  0.57   0.47  3.3 0.96
Q12 2571  0.67  0.62   0.50  2.8 0.92
Q16 2571  0.70  0.66   0.53  3.1 0.92
Q20 2571  0.55  0.45   0.35  2.4 1.04
Q21 2571  0.70  0.66   0.54  2.8 0.98

Non missing response frequency for each item
       1    2    3    4    5 miss
Q01 0.02 0.07 0.29 0.52 0.11    0
Q03 0.03 0.17 0.34 0.26 0.19    0
Q04 0.05 0.17 0.36 0.37 0.05    0
Q05 0.04 0.18 0.29 0.43 0.06    0
Q12 0.09 0.23 0.46 0.20 0.02    0
Q16 0.06 0.16 0.42 0.33 0.04    0
Q20 0.22 0.37 0.25 0.15 0.02    0
Q21 0.09 0.29 0.34 0.26 0.02    0
```

Output 17.15

Just to illustrate the importance of reverse-scoring items before running reliability analysis, Output 17.15 shows the reliability analysis for the fear of statistics subscale but done on the original data (i.e., without item 3 being reverse scored by using the *keys* option). Note that the overall α is considerably lower (.61 rather than .82). Also, note that this item has a negative item–total correlation (which is a good way to spot if you have a potential reverse-scored item in the data that hasn't been reverse scored). Finally, note that for item 3, the α if item deleted is .8. That is, if this item were deleted then the reliability would improve from about .6 to about .8. This, I hope, illustrates that failing to reverse-score items that have been phrased oppositely to other items on the scale will mess up your reliability analysis.

Moving swiftly on to the fear of maths subscale (items 8, 11 and 17), Output 17.16 shows the output from the analysis. As with the previous two subscales, the overall α is around .8, which indicates good reliability. The values of alpha if the item were deleted indicate that none of the items here would increase the reliability if they were deleted because all values in this column are less than the overall reliability of .82. The values of the corrected item–total correlations (*r.drop*) are again all above .3, which is good,

```
Reliability analysis
Call: alpha(x = mathFear)

  raw_alpha std.alpha G6(smc) average_r mean   sd
      0.82      0.82    0.75       0.6  3.7 0.75

 Reliability if an item is dropped:
    raw_alpha std.alpha G6(smc) average_r
Q08      0.74      0.74    0.59      0.59
Q11      0.74      0.74    0.59      0.59
Q17      0.77      0.77    0.63      0.63

 Item statistics
       n    r r.cor r.drop mean   sd
Q08 2571 0.86  0.76   0.68  3.8 0.87
Q11 2571 0.86  0.75   0.68  3.7 0.88
Q17 2571 0.85  0.72   0.65  3.5 0.88

Non missing response frequency for each item
       1    2    3    4    5 miss
Q08 0.03 0.06 0.19 0.58 0.15    0
Q11 0.02 0.06 0.22 0.53 0.16    0
Q17 0.03 0.10 0.27 0.52 0.08    0
```

Output 17.16

Finally, if you run the analysis for the final subscale of peer evaluation, you should get Output 17.17. Unlike the previous subscales, the overall α is quite low at .57 and although this is in keeping with what Kline says we should expect for this kind of social science data, it is well below the other scales. The values of alpha if the item is dropped indicate that none of the items here would increase the reliability if they were deleted because all values in this column are less than the overall reliability of .57. The values of *r.drop* are all around .3, and in fact for item 23 the value is below .3. This indicates fairly bad internal consistency and identifies item 23 as a potential problem. The scale has five items, compared to seven, eight and three on the other scales, so its reduced reliability is not going to be dramatically affected by the number of items (in fact, it has more items than the fear of maths subscale). If you look at the items on this subscale, they cover quite diverse themes of peer evaluation, and this might explain the relative lack of consistency. This might lead us to rethink this subscale.

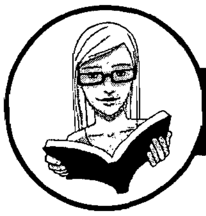
```
Reliability analysis
Call: alpha(x = peerEvaluation)
  raw_alpha std.alpha G6(smc) average_r mean   sd
      0.57      0.57    0.53      0.21  3.4 0.65

 Reliability if an item is dropped:
    raw_alpha std.alpha G6(smc) average_r
Q02      0.52      0.52    0.45      0.21
Q09      0.48      0.48    0.41      0.19
Q19      0.52      0.53    0.46      0.22
Q22      0.49      0.49    0.43      0.19
Q23      0.56      0.57    0.50      0.25

 Item statistics
       n    r r.cor r.drop mean   sd
Q02 2571 0.61  0.45   0.34  4.4 0.85
Q09 2571 0.66  0.53   0.39  3.2 1.26
Q19 2571 0.60  0.42   0.32  3.7 1.10
Q22 2571 0.64  0.50   0.38  3.1 1.04
Q23 2571 0.53  0.31   0.24  2.6 1.04

Non missing response frequency for each item
       1    2    3    4    5 miss
Q02 0.01 0.04 0.08 0.31 0.56    0
Q09 0.08 0.28 0.23 0.20 0.20    0
Q19 0.02 0.15 0.22 0.33 0.29    0
Q22 0.05 0.26 0.34 0.26 0.10    0
Q23 0.12 0.42 0.27 0.12 0.06    0
```

Output 17.17



CRAMMING SAM'S TIPS **Reliability**

- Reliability is really the consistency of a measure.
- Reliability analysis can be used to measure the consistency of a questionnaire.
- Remember to deal with reverse-scored items. Use the keys option when you run the analysis.
- Run separate reliability analyses for all subscales of your questionnaire.
- Cronbach's α indicates the overall reliability of a questionnaire and values around .8 are good (or .7 for ability tests and such like).
- The *raw alpha* when an item is dropped tells you whether removing an item will improve the overall reliability: values greater than the overall reliability indicate that removing that item will improve the overall reliability of the scale. Look for items that dramatically increase the value of α.
- If you do remove items, rerun your factor analysis to check that the factor structure still holds!

17.9. Reporting reliability analysis ②

You can report the reliabilities in the text using the symbol α and remembering that because Cronbach's α can't be larger than 1 then we drop the zero before the decimal place (if we are following APA format):

- The fear of computers, fear of statistics and fear of maths subscales of the RAQ all had high reliabilities, all Cronbach's $\alpha = .82$. However, the fear of negative peer evaluation subscale had relatively low reliability, Cronbach's $\alpha = .57$.

However, the most common way to report reliability analysis when it follows a factor analysis is to report the values of Cronbach's α as part of the table of factor loadings. For example, in Table 17.1 notice that in the last row of the table I have quoted the value of Cronbach's α for each subscale in turn.

What have I discovered about statistics? ②

This chapter has made us tiptoe along the craggy rock face that is factor analysis. This is a technique for identifying clusters of variables that relate to each other. One of the difficult things with statistics is realizing that they are subjective: many books (this one included, I suspect) create the impression that statistics are like a cook book and if you follow the instructions you'll get a nice tasty chocolate cake (yum!). Factor analysis perhaps more than any other test in this book illustrates how incorrect this is. The world of statistics is full of arbitrary rules that we probably shouldn't follow (.05 being the classic example) and nearly all of the time, whether you realize it or not, we should act upon our own discretion. So, if nothing else, I hope you've discovered enough to give you sufficient discretion about factor analysis to act upon! We saw that the first stage of factor analysis is to scan your variables to check that they relate to each other to some degree but not too strongly. The factor analysis itself has several stages: check some initial issues (e.g., sample size adequacy), decide how many factors to retain, and finally decide which items load on which factors (and try to make sense of the meaning of the factors). Having done all that, you can consider whether the items you have are reliable measures of what you're trying to measure.

We also discovered that at the age of 23 I took it upon myself to become a living homage to the digestive system. I furiously devoured articles and books on statistics (some of them I even understood), I mentally chewed over them, I broke them down with the stomach acid of my intellect, I stripped them of their goodness and nutrients, I compacted them down, and after about two years I forced the smelly brown remnants of those intellectual meals out of me in the form of a book. I was mentally exhausted at the end of it; 'It's a good job I'll never have to do that again', I thought.

R packages used in this chapter

corpcor	psych
GPArotation	

R functions used in this chapter

abs()	hist()
alpha()	kmo()
as.matrix()	mean()
c()	nrow()
cat()	plot()
cbind()	polychor()
cor()	principal()
cortest.bartlett()	print.psych()
det()	residual.stats()
factor.model()	round()
factor.residuals()	sqrt()
factor.structure()	sum()
ggplot2()	upper.tri()

Key terms that I've discovered

Alpha factoring	Kaiser–Meyer–Olkin (KMO) measure of sampling adequacy
Bartlett's test of sphericity	Latent variable
Common variance	Oblique rotation
Communality	Orthogonal rotation
Component matrix	Pattern matrix
Confirmatory factor analysis (CFA)	Principal components analysis (PCA)
Cronbach's α	Promax
Direct oblimin	Quartimax
Extraction	Random variance
Factor	Rotation
Factor analysis	Scree plot
Factor loading	Singularity
Factor matrix	Split-half reliability
Factor scores	Structure matrix
Factor transformation matrix,	Unique variance
Kaiser's criterion	Varimax

Smart Alex's tasks

- **Task 1**: The University of Sussex is constantly seeking to employ the best people possible as lecturers (no, really, it is). Anyway, they wanted to revise a questionnaire based on Bland's theory of research methods lecturers. This theory predicts that good research methods lecturers should have four characteristics: (1) a profound love of statistics; (2) an enthusiasm for experimental design; (3) a love of teaching; and (4) a complete absence of normal interpersonal skills. These characteristics should be related (i.e., correlated). The 'Teaching of Statistics for Scientific Experiments' (TOSSE) already existed, but the university revised this questionnaire and it became the 'Teaching of Statistics for Scientific Experiments – Revised' (TOSSE–R). They gave this questionnaire to 239 research methods lecturers around the world to see if it supported Bland's theory. The questionnaire is in Figure 17.9, and the data are in TOSSE.R.dat. Conduct a factor analysis (with appropriate rotation) to see the factor structure of the data. ②

SD = Strongly Disagree, D = Disagree, N = Neither, A = Agree, SA = Strongly Agree

		SD	D	N	A	SA
1	I once woke up in the middle of a vegetable patch hugging a turnip that I'd mistakenly dug up thinking it was Roy's largest root	○	○	○	○	○
2	If I had a big gun I'd shoot all the students I have to teach	○	○	○	○	○
3	I memorize probability values for the F-distribution	○	○	○	○	○
4	I worship at the shrine of Pearson	○	○	○	○	○
5	I still live with my mother and have little personal hygiene	○	○	○	○	○
6	Teaching others makes me want to swallow a large bottle of bleach because the pain of my burning oesophagus would be light relief in comparison	○	○	○	○	○
7	Helping others to understand sums of squares is a great feeling	○	○	○	○	○
8	I like control conditions	○	○	○	○	○
9	I calculate three ANOVAs in my head before getting out of bed every morning	○	○	○	○	○
10	I could spend all day explaining statistics to people	○	○	○	○	○
11	I like it when people tell me I've helped them to understand factor rotation	○	○	○	○	○
12	People fall asleep as soon as I open my mouth to speak	○	○	○	○	○
13	Designing experiments is fun	○	○	○	○	○
14	I'd rather think about appropriate dependent variables than go to the pub	○	○	○	○	○
15	I soil my pants with excitement at the mere mention of factor analysis	○	○	○	○	○
16	Thinking about whether to use repeated or independent measures thrills me	○	○	○	○	○
17	I enjoy sitting in the park contemplating whether to use participant observation in my next experiment	○	○	○	○	○
18	Standing in front of 300 people in no way makes me lose control of my bowels	○	○	○	○	○
19	I like to help students	○	○	○	○	○
20	Passing on knowledge is the greatest gift you can bestow on an individual	○	○	○	○	○
21	Thinking about Bonferroni corrections gives me a tingly feeling in my groin	○	○	○	○	○
22	I quiver with excitement when thinking about designing my next experiment	○	○	○	○	○
23	I often spend my spare time talking to the pigeons ... and even they die of boredom	○	○	○	○	○
24	I tried to build myself a time machine so that I could go back to the 1930s and follow Fisher around on my hands and knees licking the floor on which he'd just trodden	○	○	○	○	○
25	I love teaching	○	○	○	○	○
26	I spend lots of time helping students	○	○	○	○	○
27	I love teaching because students have to pretend to like me or they'll get bad marks	○	○	○	○	○
28	My cat is my only friend	○	○	○	○	○

FIGURE 17.9 The Teaching of Statistics for Scientific Experiments – Revised (TOSSE–R)

- **Task 2:** Dr Sian Williams (University of Brighton) devised a questionnaire to measure organizational ability. She predicted five factors to do with organizational ability: (1) preference for organization; (2) goal achievement; (3) planning approach; (4) acceptance of delays; and (5) preference for routine. These dimensions are theoretically independent. Williams's questionnaire (Figure 17.10) contains 28 items using a 7-point Likert scale (1 = strongly disagree, 4 = neither, 7 = strongly agree). She gave it to 239 people. Run a principal components analysis on the data in **Williams.dat.** ②

Answers can be found on the companion website.

FIGURE 17.10 Williams's organizational ability questionnaire

1	I like to have a plan to work to in everyday life
2	I feel frustrated when things don't go to plan
3	I get most things done in a day that I want to
4	I stick to a plan once I have made it
5	I enjoy spontaneity and uncertainty
6	I feel frustrated if I can't find something I need
7	I find it difficult to follow a plan through
8	I am an organized person
9	I like to know what I have to do in a day
10	Disorganized people annoy me
11	I leave things to the last minute
12	I have many different plans relating to the same goal
13	I like to have my documents filed and in order
14	I find it easy to work in a disorganized environment
15	I make 'to do' lists and achieve most of the things on it
16	My workspace is messy and disorganized
17	I like to be organized
18	Interruptions to my daily routine annoy me
19	I feel that I am wasting my time
20	I forget the plans I have made
21	I prioritize the things I have to do
22	I like to work in an organized environment
23	I feel relaxed when I don't have a routine
24	I set deadlines for myself and achieve them
25	I change rather aimlessly from one activity to another during the day
26	I have trouble organizing the things I have to do
27	I put tasks off to another day
28	I feel restricted by schedules and plans

Further reading

Cortina, J. M. (1993). What is coefficient alpha? An examination of theory and applications. *Journal of Applied Psychology*, *78*, 98–104. (A very readable paper on Cronbach's α.)

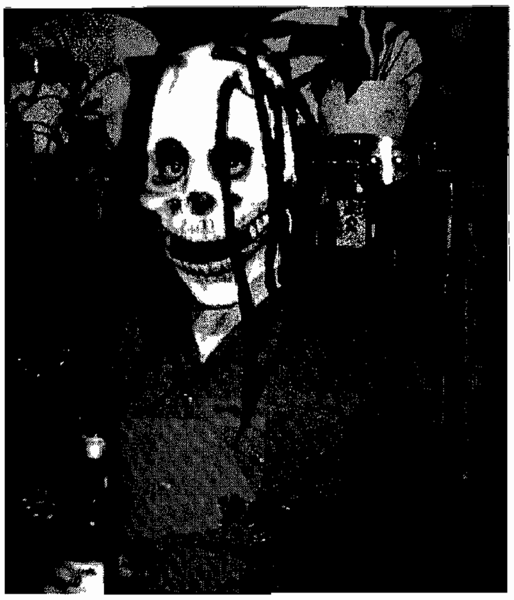
Dunteman, G. E. (1989). *Principal components analysis.* Sage University Paper Series on Quantitative Applications in the Social Sciences, 07-069. Newbury Park, CA: Sage. (This monograph is quite high level but comprehensive.)

Pedhazur, E., & Schmelkin, L. (1991). *Measurement, design and analysis*. Hillsdale, NJ: Erlbaum. (Chapter 22 is an excellent introduction to the theory of factor analysis.)
Tabachnick, B. G. & Fidell, L. S. (2007). *Using multivariate statistics* (5th ed.). Boston: Allyn & Bacon. (Chapter 13 is a technical but wonderful overview of factor analysis.)

Interesting real research

Nichols, L. A., & Nicki, R. (2004). Development of a psychometrically sound internet addiction scale: A preliminary step. *Psychology of Addictive Behaviors*, *18*(4), 381–384.

18 Categorical data


IGURE 18.1
ʻlidway through
/riting the second
dition of my
PSS book, things
ad gone a little
trange

18.1. What will this chapter tell me? ①

We discovered in the previous chapter that I wrote a book. An earlier edition of this book, which focused on SPSS. There are a lot of good things about writing books. The main benefit is that your parents are impressed. Well, they're not *that* impressed actually, because they think that a good book sells as many copies as *Harry Potter* and that people should queue outside bookshops for the latest enthralling instalment of *Discovering Statistics* .... My parents are, consequently, quite baffled about how this book is seen as reasonably successful, yet I don't get invited to dinner by the Queen. Nevertheless, given that my family don't really understand what I do, books are tangible proof that I do *something*. The size