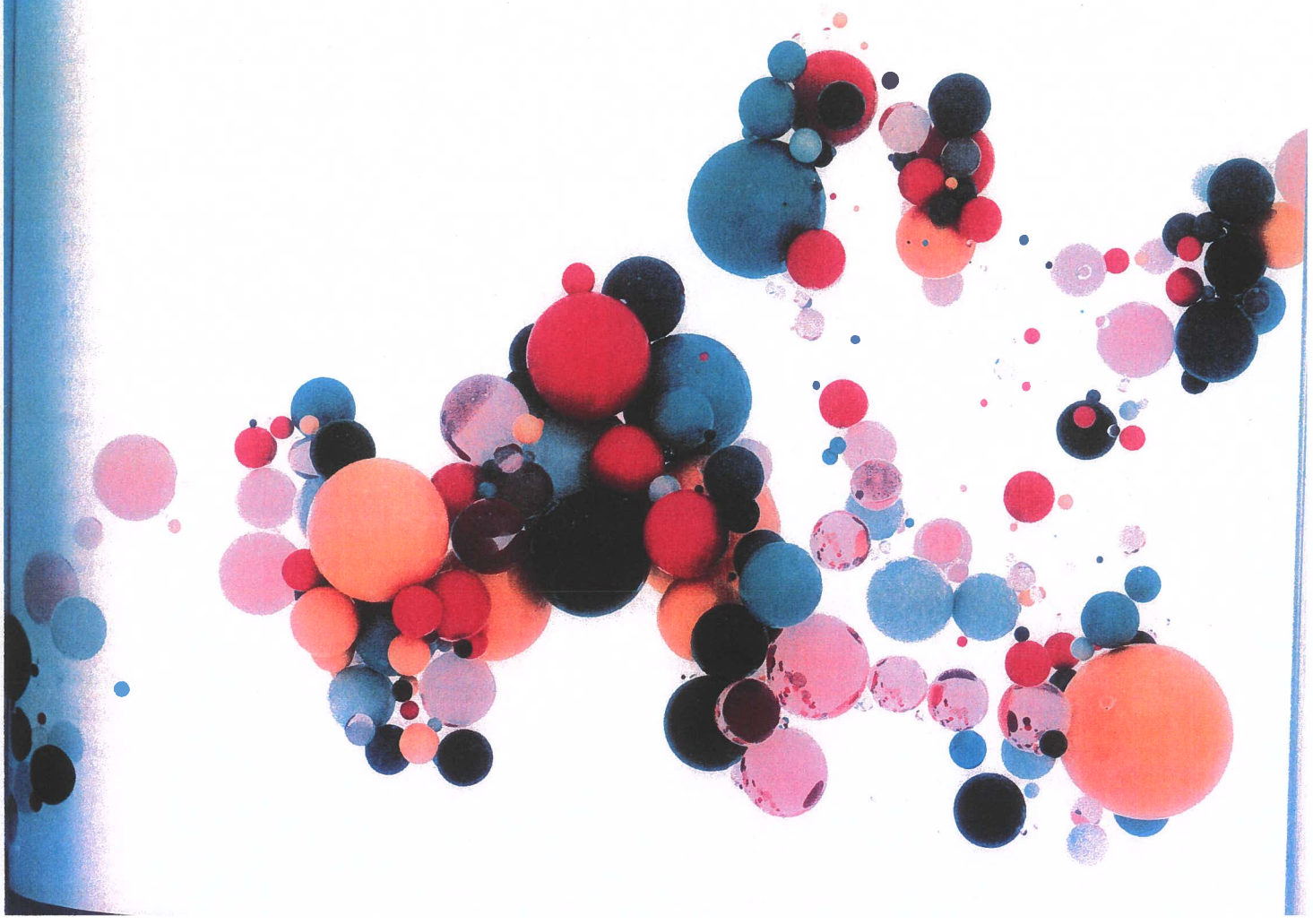
13
Factor Analysis



In this chapter, we make use of the following packages:

- `astatur` - contains the datasets used in this chapter
- `tidyverse` - functions for data management (installs `dplyr`, `tidyr`, and more)
- `psych` - functions for factor analysis

These packages must be installed and activated to run the code provided in this chapter. We can install these by typing the following commands:

```
packages <- c("tidyverse", "psych",
              "devtools")
install.packages(packages)
devtools::install_github("ihrke/astatur")
```

Learning outcomes

- Understand the purpose of factor analysis
- Explain the stages of factor analysis
- Know the difference between principal component analysis and other factor extraction methods
- Understand and interpret factor analysis using R
- Obtain estimated and generated factor scores and perform reliability tests

In this chapter, we explain exploratory factor analysis (EFA), or just factor analysis, a technique which in practice is often used for data reduction purposes in the social sciences. While we focus exclusively on EFA in the present chapter, we will present confirmatory factor analysis (CFA) in Chapter 14 as a special case of structural equation modelling. Where appropriate, we will clarify the major differences between factor analysis and principal component analysis (PCA; another statistical technique used for data reduction). After explaining the details (extraction, loadings, rotation, etc.) of factor analysis, we illustrate how we can apply it to a real-life dataset using R. As reduced datasets containing factors representing different subsets of variables are often used as dependent or independent variables in subsequent regression analyses in many applications in the social sciences, we also explain the procedures (factor score computation, reliability tests, etc.) necessary prior to the subsequent use of these factors/components.

13.1 What Is Factor Analysis?

Factor analysis is a statistical technique used to detect a smaller set of underlying so-called factors that explain the covariance/correlation pattern among a larger set of observed variables. Alternative terms for factors are unobserved variables, hypothetical variables, latent variables, and constructs, while alternative terms for observed variables are items, indicators,

manifest variables, and measured variables. Conceptually, each factor would correspond to a combination of a subset of observed variables that are relatively highly correlated. To further clarify the notion of factor analysis, let us use a hypothetical example. Suppose that a random sample of people have responded to the following statements (on a scale from 1 = *totally disagree* to 5 = *totally agree*) included in a questionnaire distributed by an environmental psychologist:

`Var1` - Most of my friends think I should use environmentally friendly products
`Var2` - Most of my neighbours think I should use environmentally friendly products
`Var3` - Most of my colleagues think I should use environmentally friendly products
`Var4` - I feel a moral obligation to buy environmentally friendly products
`Var5` - I feel a moral obligation to recycle household waste
`Var6` - I feel a moral obligation to buy products made with recycled ingredients

In this case, responses given to each of these statements would represent observed variables represented by `Var1` to `Var6` in the researcher's dataset. Suppose further that we discover that `Var1`, `Var2`, and `Var3` are highly correlated and that `Var4`, `Var5`, and `Var6` are correlated (see Figure 13.1). We can now assume that there are two unobserved, hypothetical concepts, say, `SOCIAL NORMS` and `PERSONAL NORMS`, that are underlying the responses for `Var1` to `Var3` and `Var4` to `Var6`, respectively, and hence are responsible for the high correlations among them. Under that assumption, it can be useful to work with the hypothetical factors instead of the raw variable scores for practical and theoretical reasons.

	Var1	Var2	Var3	Var4	Var5	Var6
Var1	**0.6844**					
Var2	0.8219	**0.8115**				
Var3	0.7391	0.8555	**0.7392**			
Var4	0.2295	0.1975	0.2194	**0.3440**		
Var5	0.1645	0.1658	0.1906	0.5492	**0.3677**	
Var6	0.1315	0.1331	0.1631	0.3826	0.4427	**0.2275**

Figure 13.1 Initial correlation matrix with communalities in the diagonal

Schematically, the above example factor model can be drawn as in Figure 13.2 in which `SOCIAL NORMS` and `PERSONAL NORMS` are the two factors assumed to account for the variance in all six items (`Var1` to `Var6`). Note that, in this example, we do not assume correlation between the two factors (i.e. the factors are assumed to be orthogonal to each other). If we had assumed a correlation between the factors, we would have drawn a curve linking the two factors together. However, the two factors cannot completely account for all of the variance in the six items just as regression models never result in a perfect fit. Hence, we use an error term to capture variance from other causes besides the factors in each item (the small `e1`, `e2`, etc., to the right of Figure 13.2). Perhaps surprisingly, factor analysis can simply be presented in the form of a regression equation as well. To illustrate this idea, we rewrite `Var1`, . . ., `Var6`

as $Y_1, \ldots, Y_6$ and the two factors as X_1 and X_2. That means we treat the (unknown) factors as independent and the observed item scores as dependent variables as depicted in Figure 13.2. The resulting equation resembles a multiple regression equation (generalized below) including two predictors for any of the six observed variables:

$$Y_{ji} = \beta_{0j} + \beta_{1j}X_{1i} + \beta_{2j}X_{2i} + \varepsilon_{ji}. \tag{13.1}$$

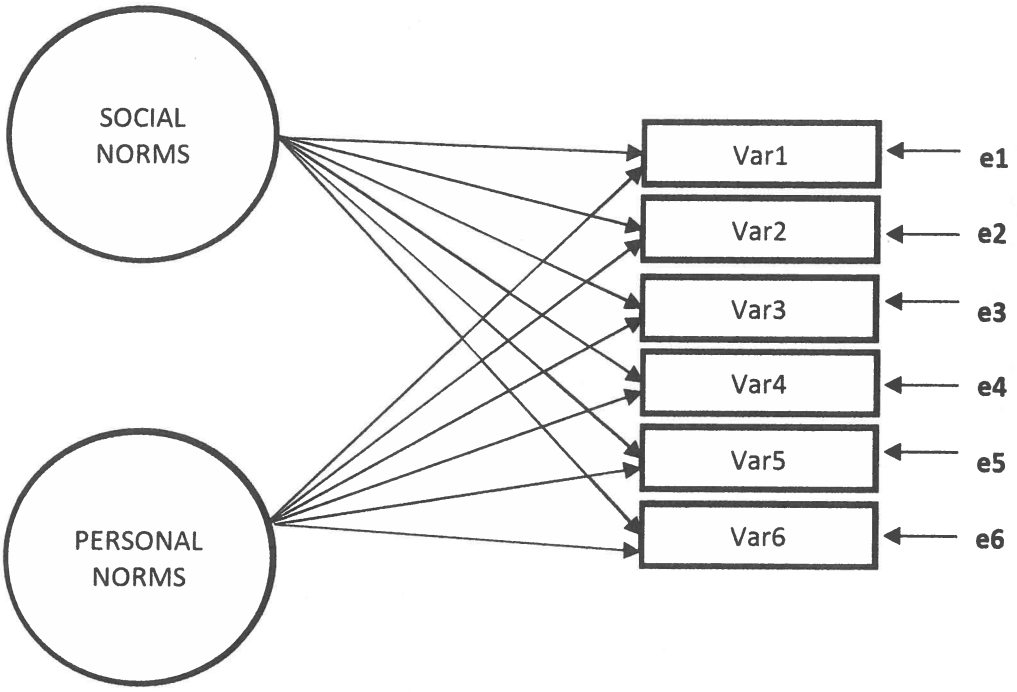


Figure 13.2 Schematic representation of a factor model

Since we normally work with standardized variables in factor analysis, the constant/intercept will be 0, thus our equation should be rewritten as:

$$Y_{ji} = \beta_{1j}X_{1i} + \beta_{2j}X_{2i} + \varepsilon_{ji}. \tag{13.2}$$

Note that the indices, *j* and *i*, represent the *j*th variable and the *i*th measurement. This means that $Y_{2,4}$ is the measurement of the fourth respondent to Question 2 (`Var2`). We can now see that the factors X_1 and X_2 are constant across the variables *j* but change for each respondent *i*. Equation (13.2) also shows how the value of each of the observed variables is estimated and that the influence of the two factors on each observed variable is determined by their respective weights represented by the regression coefficients as well as the error term. Regression coefficients and error term have their corresponding names (loadings and uniqueness) in the realm of factor analysis, which we will come back to later in the chapter.

13.1.1 What is factor analysis used for?

As the definition of the factor analysis provided above indicates, the method can be used for various applications. First, we can use factor analysis to reduce a large number of variables down to a meaningful and manageable number of factors that reflect most of the information

contained in the original variables. Second, factor analysis is used to examine the dimensionality of a set of variables. Here, the researcher would be interested in how many factors provide the most suitable description of the raw dataset. The optimal number of factors would then correspond to the dimensionality of the dataset. If there is more than one dimension underlying the variables, then factor analysis can reveal how many and which variables belong to which dimensions. Third, factor analysis is used to assess some of the psychometric properties of a multidimensional scale. Fourth and relatedly, factor analysis is used in the early stages of a scale development. The just-mentioned uses of factor analysis are somewhat overlapping as factor analysis can serve several purposes in a single study.

13.2 The Factor Analysis Process

Regardless of the aim of the analysis, the factor analysis process involves four main steps: (1) determining the number of factors, (2) extracting the factors, (3) rotating the factors, and (4) refining and interpreting the factors. Figure 13.3 depicts factor analysis as a circular process. The reason for this is that many researchers in practice move back and forth between the four main steps in an attempt to find the best factor solution.

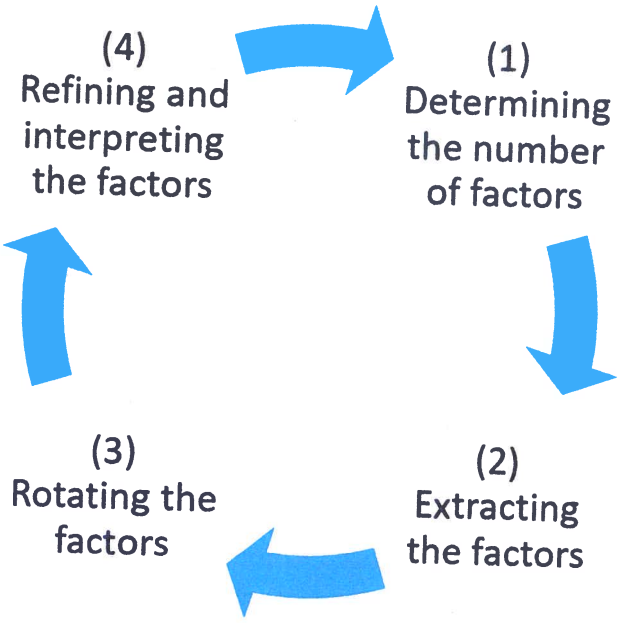


Figure 13.3 Circular process of factor analysis

13.2.1 Determining the number of factors

As the first step of the factor analysis process, we have to make a decision as to the number of factors to retain. There is unfortunately no clear-cut answer to this question in the multivariate statistics literature. There are, however, some good reasons for using a combination of the following criteria when deciding the number of factors: eigenvalue rule, scree test, parallel analysis, and theoretical sensitivity.

Eigenvalue rule

In PCA (a variant of factor analysis), factors associated with eigenvalues greater than 1 are generally recommended to be retained for interpretation. The idea behind this criterion is that the factor should explain at least as much variance as one observed variable contributes, a situation represented by 1s in the diagonal of the correlation matrix. Since in factor analysis we insert communalities (which are less than 1, see below for details) instead of 1s in the diagonal for factor analysis (*principal axis* [PA] or *iterated* PA), the rule of eigenvalues greater than 1 should accordingly be adjusted. One such rule is to choose eigenvalues greater than the average of the initial communalities (Afifi et al., 2012).

Scree test

The scree test is about examining a curve, produced after factor extraction, containing the eigenvalues on the y-axis and the factors on the x-axis. The idea of the scree test is that the factors with highest eigenvalues carry most information and that the factors at the tail of the curve represent mostly random error variance (trivial eigenvalues), and consequently, the factor solution just before the curve starts levelling off should be chosen (Kachigan, 1991).

Parallel analysis

Parallel analysis is about estimating the same factor model as the original one but using randomly simulated data instead of real data. For that to work, the simulated data must resemble the original data in terms of both the number of variables and observations. Eigenvalues obtained from such simulated datasets are subsequently averaged (e.g. after repeating the procedure 25 times on different randomly generated datasets) and contrasted with their counterparts from the original data. If the eigenvalue of a factor from the original data proves to be larger than the average of the eigenvalues from the simulated data, the factor can be retained. Instead of relying on the mean across random datasets, some suggest using the 95th percentile of the distribution of eigenvalues from the random datasets. Otherwise, the factor would be considered no more substantial than a random factor and thus discarded (see Matsunaga, 2010). Although parallel analysis is generally considered the best procedure for deciding the number of factors, there are still, as noted by Fabrigar and Wegener (2011), many conditions when its performance is untested. Thus, we recommend that parallel analysis is used in tandem with the other criteria described here.

Theoretical sensitivity

Despite the exploratory nature of factor analysis, the researcher should still employ theoretical sensitivity (subject knowledge as well as common sense) when examining the number of factors to retain. In an EFA, different factor solutions (two factors, three factors, etc.) may be considered. The most salient point is that the competing factor solutions should make theoretical/conceptual sense. Depending solely on statistical criteria may in some cases be misleading.

13.2.2 Extracting the factors

Factor extraction is the second step in the factor analysis process. The starting point of this step of the analysis is a correlation matrix including all the correlations among pairs of observed variables (see Figure 13.1). In a raw correlation matrix, the diagonal elements are always 1 because a variable is always perfectly correlated with itself. However, in a factor analysis context, the diagonal entries in the matrix can be adapted to suit the needs of the analysis. The values in the diagonal of the correlation matrix constitute essentially, at least mathematically, the only difference among many of the different factor extraction methods.[1] The main factor extraction methods are principal (axis) (PA), iterated principal (axis) factor (IPA), PCA, and maximum-likelihood (ML) factor. In this chapter, we will focus on PA, IPA, and PCA, and we will present ML in Chapter 14 when we review CFA.

The extraction method PA inserts estimates of the commonly shared variance (also called communality) in the diagonal of the starting correlation matrix instead of 1s (see Figure 13.1). Communality values are obtained by estimating the squared multiple correlation (SMC) of each variable with all the other variables in the matrix obtained by regressing each variable on the remaining variables. The reason why PA uses communalities in the diagonals is that it assumes that some of the variance in the variables is caused by some other sources which should ideally be removed from the analysis. All factor analysis methods apart from PCA make this assumption. This is also the main difference between all of the factor analysis extraction methods (PA, IPA, and ML) and the PCA method. While the PCA method uses 1s in the diagonal of the correlation matrix, the other measures use the communalities. As a consequence, we can assert that PCA analyses variance (1s in the diagonals), whereas the PA, IPA, and ML methods analyse covariance (communalities in the diagonals; see also Tabachnick and Fidell, 2014).

Furthermore, the unique variance is in theory further divided into specific or systematic variance components and measurement error (random variance). For example, a part of the specific variance could be caused by biased wording affecting the responses of a person consistently (i.e. the effect of the wording would be consistently present were we to repeat filling out the questionnaire at a second point in time). However, such effects are specific to that item in that the biased wording would not necessarily influence responses to other items in the battery (Fabrigar and Wegener, 2011). The measurement error variance, on the other hand, can be caused by ambiguous wording that can be interpreted differently at two points in time by the same person depending on the person's psychological state (Fabrigar and Wegener, 2011) or can be due to other transient environmental factors. Even though there is such a conceptual difference, factor analysis in practice does not distinguish between specific variance and measurement error variance, which in tandem are encompassed by the term *unique variance* in the computations.

Estimating and then inserting the communalities in the diagonals of the correlation matrix was the first task in factor extraction. The next task is to compute eigenvalues and eigenvectors from this correlation matrix which are then used to compute factor loadings (see P. Kline, 1994). Eigenvectors are sets of weights (w) that generate factors with the largest possible eigenvalues, while eigenvalues (e) are variances captured by these factors. The factor extraction proceeds as follows (see P. Kline, 1994):

1. A series of eigenvectors (and their corresponding eigenvalues) are computed repeatedly from the initial correlation matrix (with communalities in the diagonals) until the solution converges (i.e. additional vectors are nearly identical to the last one); this last vector and its eigenvalue are the basis of the first factor. P. Kline (1994) explains how eigenvectors and eigenvalues are computed manually but statistical software such as R implements these calculations for us.
2. The common variance that the first factor explains/captures is then subtracted from the initial correlation matrix, resulting in a residual matrix (i.e. diagonals are now less than those in the initial matrix).
3. To extract the second factor, the same computation procedure described in Step 1 takes place once again, but this time it is based on the residual matrix instead of the initial correlation matrix.
4. The variance that the second factor explains/captures is also subtracted from the residual matrix, resulting in another reduced residual matrix.
5. To extract a further factor, the same procedure as in Step 1 takes place again using this reduced residual matrix.
6. And so on.

Based on the correlation matrix in Figure 13.1, R can compute the necessary eigenvectors and eigenvalues by adopting the PA extraction method. Figure 13.4, for instance, shows eigenvectors associated with the first two factors that have the largest eigenvalues. There will be eigenvectors and eigenvalues computed for each factor. However, here and generally, common practice is to focus only on those that capture most variance. Figure 13.4 also shows the formula used to compute the factor loadings. Here, each eigenvector for a variable is multiplied by the square root of the factor eigenvalue to obtain the factor loading for the variable.

Eigenvectors (*w*)	
Factor 1	**Factor 2**
0.5127617	-0.1951717
0.5592738	-0.2552949
0.5344650	-0.1897104
0.2399653	0.5371445
0.2218162	0.5926331
0.1784647	0.4701110

Eigenvalues (e)	
Factor 1	2.575541
Factor 2	1.030058

$$l = w\sqrt{e}$$

$= 0.513\sqrt{2.576} = -0.196\sqrt{1.030}$

$= 0.560\sqrt{2.576} = -0.255\sqrt{1.030}$

etc.

	Factor loadings (*l*)	
	Factor1	**Factor2**
Var1	0.8229093	-0.1980738
Var2	0.8975704	-0.2590631
Var3	0.8577443	-0.1925387
Var4	0.3850864	0.5451761
Var5	0.3559512	0.6014738
Var6	0.2863748	0.4771296

Figure 13.4 Overview of eigenvectors, eigenvalues, and unrotated factor loadings

As can be seen in Figure 13.4, the sum of the two eigenvalues (2.576 + 1.030 = 3.606) is the total amount of common variance (communality) these two factors explain together. This number should optimally not be larger than the sum of the diagonals in the correlation matrix (Figure 13.1) that we started with. The sum of the diagonals (3.1743) of the initial matrix in our current example is, however, somewhat less than the sum of the eigenvalues. The reason for this not unusual difference is the fact that the SMCs that we inserted in the diagonals in the initial matrix as the estimates of the communalities are not perfectly accurate (Hatcher, 2006). As this situation occurs often using principal factors, factor analysts prefer to use an iterative procedure, the IPA method. While PA starts and completes the factor extraction process (Steps 1–6 above) with one set of estimated communalities (SMCs), IPA replaces these estimated communalities by new estimates (h^2) emerging from the factor extraction process during each iteration until the difference between the two communalities (last inserted and last estimated) is minimized (Lattin et al., 2003, p. 136). In a nutshell, PA is a factor extraction process run only once, while IPA is run several times (usually around 50 iterations).

Regardless of the factor solution from PA, IPA, or PCA, factor loadings would reflect correlations between observed variables and their respective factors unless factors are correlated. Factors can be assumed as correlated when rotating the factor solution using a type of oblique rotation such as *promax* or *oblimin* – see below. As a result, if we square these correlations, the results will show us how much variance of each observed variable each factor explains. For instance, take the factor loading of `Var5` on factor 1, from the PA solution above, which is 0.3560. Squaring this gives 0.1267. This means that factor 1 explains 12.67% of the variance in `Var5`. Furthermore, we also see that the factor loading of `Var5` on factor 2 is 0.6015. Squaring this gives 0.3618, meaning that factor 2 explains 36.18% of the variance in `Var5`. Summing 12.67 and 36.18, we find that 48.85% of the variance in `Var5` is accounted for by factor 1 and factor 2 together, represented by h^2. We can thus compute the unique variance, which is 100 – 48.85 = 51.15%, including both specific and measurement error variance:

$$h_i^2 = \sum_{j=1}^{M} l_{ij}^2. \tag{13.3}$$

Unique variances are provided under the column titled u^2 in the R output.

13.2.3 Rotating the factors

Having decided on the number of factors, the next task in the factor analysis process is, as depicted in Figure 13.5, to rotate the initial factor solution (the black axis) in order to obtain a more easily interpretable factor solution (the dashed axis). An easily interpretable factor solution is associated with an output that contains a factor loading (pattern) matrix with variables with the maximum possible loading (close to 1) on one factor and the minimum possible loading (close to 0) on the remaining factor(s), a situation often referred to as a 'simple structure' in the literature, as first described by Thurstone (1947).

In the unrotated solution, as presented in Figure 13.4 and further visualized in Figure 13.5, three of the variables (`Var4-Var6`) load heavily on (are close to) both factors. This is in fact a common occurrence when using unrotated solutions in factor analyses. Geometrically, this means that the distance between `Var4` to `Var6` and factor 2 (d_2) when compared with the

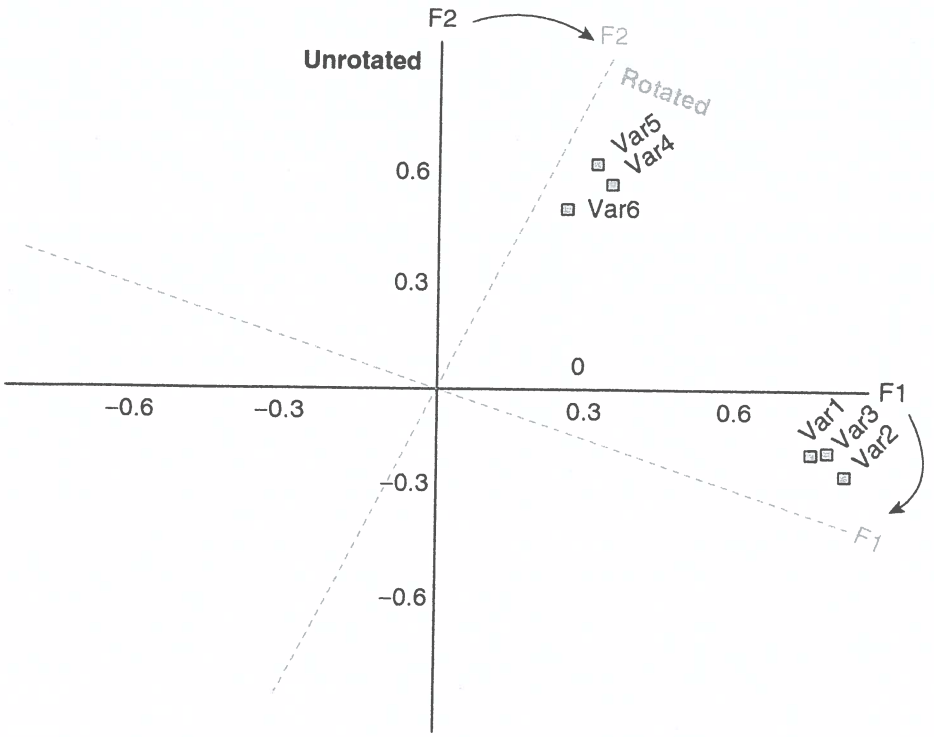


Figure 13.5 Geometrical representation of an orthogonal rotation

distance between `Var4` to `Var6` and factor 1 (d_1) is not substantial. After rotation, however, this distance ($d_2 - d_1$) becomes more apparent because `Var4` to `Var6` get closer to factor 2 but more distant from factor 1. In the rotated solution, we therefore obtain new coordinates showing that `Var4` to `Var6` load more strongly on factor 2 while loading more weakly on factor 1. Incidentally, the strong and weak loadings of `Var1` to `Var3` on factor 1 and factor 2 do not change substantially. Geometrically the closer the variables are to each other or factors, the higher the correlations are, and vice versa.

The new coordinates in this example have been estimated using the most widely used orthogonal rotation technique, namely *varimax* rotation. Varimax maximizes the variance of the squared loadings for each factor, thus polarizing loadings so that they are either high or low, making it easier to identify factors with specific variables (Hamilton, 1992, p. 261). As also shown in Figure 13.5, varimax keeps the angle between the axis of factor 1 and factor 2 at 90° indicating zero correlation between the two factors. If it were not for this restriction, the rotated axes could go right through the cloud of variables in Figure 13.5.

It should further be noted that rotation does not influence (increase or decrease) the *total variance explained* by the factors. However, the total variance explained gets distributed differently among the factors (i.e. *eigenvalues change*). Suppose that two factors explain 75% variance together, with 48% and 27% of the variance attributable to the first and second factor, respectively. After rotation, the factors would together still explain 75% variance, but now with, say, 44% and 31% of the total variance attributed to the first and second factor, respectively. Relatedly, the total amount of variance of each variable explained by the factors (i.e. the communality) would also stay the same, but the factor loadings would change after rotation. In addition to varimax, there are several other orthogonal rotation techniques such as *quartimax*, *equamax*, and *parsimax*, which are less commonly encountered in social science research.

As an alternative to orthogonal rotation, oblique rotation can be used. The idea of oblique rotation is that the restriction of a 90° angle between the factor axes (imposed by orthogonal rotation) is relaxed when rotating (see Figure 13.6). There is general support for this relaxation among social scientists as it is more realistic that factors (latent variables) measuring behavioural phenomena are somewhat correlated. As such, it is widely recommended that one by default employs an oblique rotation to be able to take this correlation into account in the factor model estimation (Harman, 1976). In the case where the factors are truly not correlated or only weakly correlated, an oblique rotation would produce a similar solution to that provided by an orthogonal rotation (Harman, 1976).

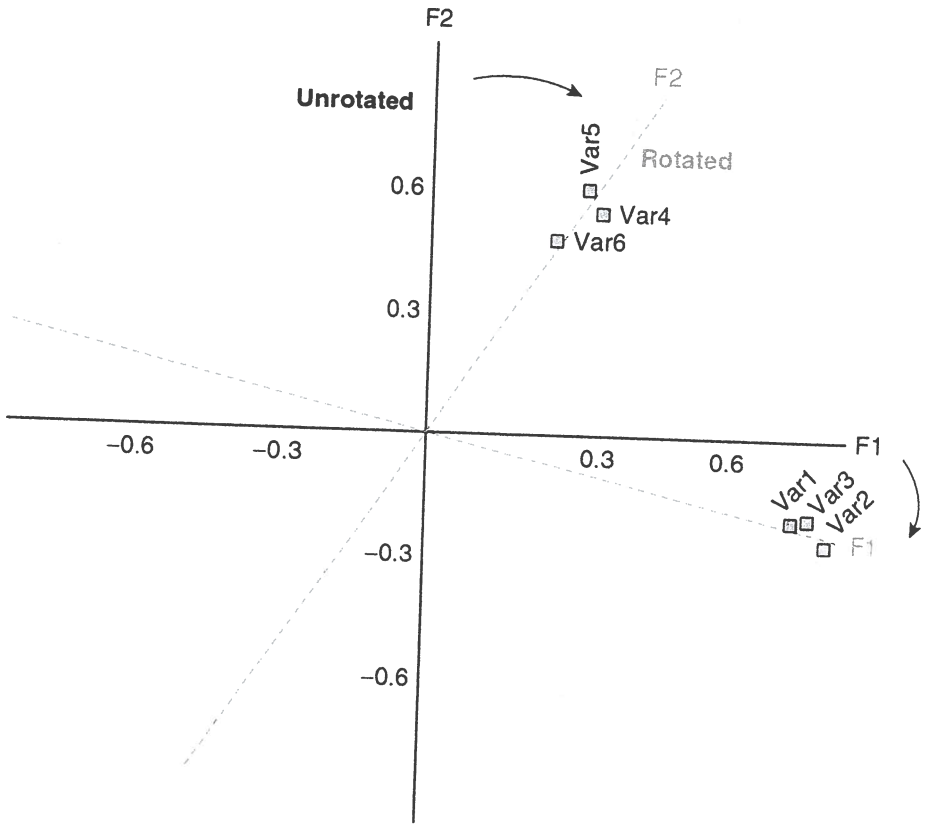


Figure 13.6 Geometrical representation of an oblique rotation

The most common oblique rotation technique is *promax*. Promax starts with an orthogonal rotation (*varimax*) in which the loadings are raised to a specific power (2, 3, or 4), and then the solution is rotated to allow for correlations among the factors (Pett et al., 2003). Raising varimax loadings/coefficients to a given power makes all the resulting loadings/coefficients closer to zero, but the effect differs across original values that are larger in magnitude (e.g. $0.87^3 = 0.66$) as opposed to smaller in magnitude (e.g. $0.25^3 = 0.016$) (Thompson, 2004). The goal here is to obtain a factor solution containing the best structure using the lowest possible power loadings and thus with the lowest correlation among the factors (Pett et al., 2003). Power values larger than 4 are generally not recommended (StataCorp, 2015, p. 662). In addition to promax, there are several other oblique rotation methods, among others *oblimin*, *oblimax*, and *quartimin*, which are less commonly used in the literature.

13.2.4 Refining and interpreting the factors

After rotating the factors, the next step is to refine and interpret them using both quantitative criteria and qualitative judgement. Qualitative judgement is about examining the factor solution on theoretical and/or conceptual grounds and, therefore, depends on the concrete situation in which factor analysis is applied. One salient quantitative criterion is the examination of factor loadings. A factor loading is a quantity measuring the strength of the relationship between an observed variable and a factor. In an orthogonal solution, a factor loading corresponds directly to the standardized coefficient (or correlation) in a bivariate/simple regression model. In an oblique solution, however, a factor loading corresponds to the partial standardized coefficient (or partial correlation) in a multiple regression model. This is because, in the oblique solution, the relationship (loading) between an observed variable and a factor is estimated having controlled for the other factor(s), as visualized in Figure 13.7.

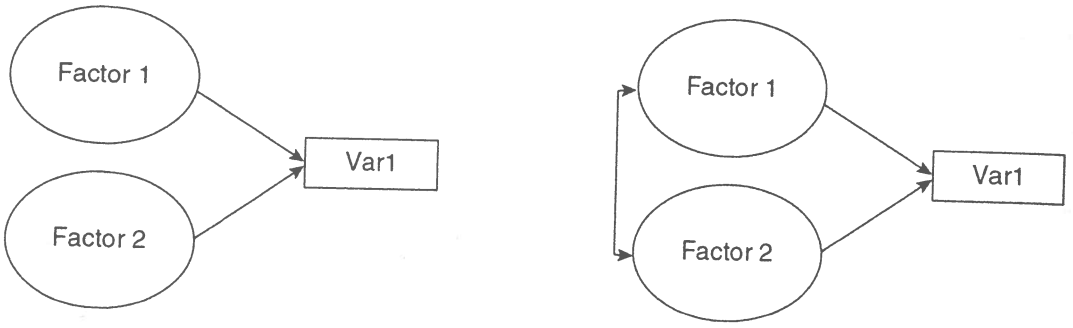


Figure 13.7 Orthogonally (left) and obliquely (right) rotated factor solutions

In line with the above explanation, the interpretation of the *factor loading matrix* (*factor pattern matrix*) produced by orthogonal or oblique rotation will be different. In the orthogonal case (see Figure 13.7), we would interpret, for instance, a loading of 0.8 of `Var1` on factor 1 as 'for every unit increase in factor 1, there will be an average increase of 0.8 in `Var1`'. This interpretation would slightly change in the oblique case. Here, we would instead say 'for every unit increase in factor 1, there will be an average increase of 0.8 in `Var1`, having controlled for factor 2'.

In addition to the pattern matrix, oblique rotation generates two other matrices: a factor correlation matrix and a factor structure matrix. The *factor correlation matrix* shows the correlations between the factors. The *factor structure matrix* contains loadings which are zero-order correlations of the observed variables with the factors. Since these loadings are not adjusted for the correlation between the factors, there is little use in interpreting these. We should thus examine and report the results from the factor pattern matrix instead of those from the factor structure matrix.

When examining the factor pattern matrix, regardless of whether orthogonal or oblique rotation has been used and based on a general consensus found in the literature (e.g. Brown, 2015; Hatcher, 2006), we suggest using a value of 0.4 as a threshold to distinguish between practically significant and non-significant loadings. This means that loadings below 0.4 would indicate a weak relationship between an observed variable and a factor. In PCA, on the other hand, the threshold should be set to 0.7. The reason is that the loadings would usually be lower in factor analysis than in PCA as a result of values lower than 1 occurring on the diagonals in factor analysis.

Variables loading weakly (i.e. lower than the threshold of 0.4 or 0.7) on all of the factors in a solution should generally be removed from the analysis. This general rule applies also to variables loading too strongly on more than one factor, as this situation would cause problems for establishing discriminant validity of the factors. Removal of the observed variables must be done sequentially, meaning that after deletion of each variable, the factor model should be re-estimated and examined before possibly removing another variable. Removal of variables should be considered only after having tried out different extraction methods as well as rotation techniques.

We further suggest that removal (as well as inclusion) of variables should be guided by the theoretical reasoning of the researcher. The researcher should examine whether variables loading strongly on a factor can actually be represented by the factor conceptually as well. The sign of the loadings is also important to examine. We recommend that variables loading negatively on factors be reversed to make the interpretation easier. While examining each variable's content and loadings on different factors, the researcher can start labelling the factors as a way to interpret the solution. These labels would usually be inspired by the relevant theory as well as common sense.

Labelling the factors completes the circular factor analysis process. Nevertheless, many social scientists wish to use the resulting factors in further analyses. That is, the resulting factors are often used as predictors (and sometimes as outcomes) in a statistical model. To do so, the factors must be given a metric and their reliabilities must be assessed.

13.3 Composite Scores and Reliability Tests

There are two main ways of computing a metric (or a composite score) for a factor: *estimating* a factor score and *generating* a factor score (i.e. a summed scale). Estimated factor scores are standardized and weighted values showing the score of each individual on the factor. As shown in Figure 13.8, an individual's estimated factor score is computed by summing the products of the standardized factor score coefficients (weights) and the standardized scores (i.e. *z*-scores) of the individual on all of the variables, a similar procedure to that used for prediction purposes in multiple regression analysis.

The above method is therefore referred to in the literature as the *regression method* for estimating factor scores. Here we will only present the regression method. However, in addition to the regression method, there is also the Bartlett method and Anderson–Rubin method (see Pett et al., 2003) which we will ignore for the present purpose. Technically, factor score coefficients (weights) are computed by multiplying the inverse of the sample correlation matrix by the factor loading (pattern) matrix.

The advantage of the *estimated factor score* is that it represents all of the variables loading on the factor, while its disadvantage is that the scores obtained are not unique values (i.e. factor indeterminacy) and thus not easily replicable across studies (Hair et al., 2013; Pett et al., 2003). Consequently, it has been suggested that the factor determinacy coefficient (see Beauducel, 2011) be examined before using estimated factors scores as variables in subsequent analyses. According to Gorsuch (1983), such a coefficient should be at least 0.90 if the factor score is to be used as a substitute for the observed variables. The factor solution is not unique due to the problems of factor rotation and communality estimation (see Sharma, 1996).

Factor score coefficients

Scoring coefficients

Variable	Factor1	Factor2
Var1	0.20845	0.04980
Var2	0.53094	-0.01758
Var3	0.26327	0.08672
Var4	0.01966	0.33599
Var5	0.00944	0.38687
Var6	0.00627	0.23817

Standardized (z-scores) values

z_Var1	z_Var2	z_Var3	z_Var4	z_Var5	z_Var6
.96042038	.9582161	1.0439712	.72305588	.76442534	.04416852

$$Y_i = \beta_1 X_{1i} + \beta_2 X_{2i} + \ldots + \beta_k X_{ki}$$

Factor score of individual 1 on factor 1

$= .20845 \times .96042 + .53094 \times .95821 + .26327 \times 1.04397$
$+ .01966 \times .72305 + .00944 \times .76442 + .00627 \times .04416$
$= 1.005$

Factor score of individual 1 on factor 2

$= .04980 \times .96042 + (-.01758) \times .95821 + .08672 \times 1.04397$
$+ .33599 \times .72305 + .38687 \times .76442 + .23817 \times .04416$
$= .6707$

Figure 13.8 Estimated factor scores

Generated factor scores are raw and unweighted values obtained for each individual by either summing or averaging only those variables loading most strongly on a factor. As opposed to their standardized and weighted counterparts, generated factor scores exclude those variables that load weakly on the factor in the computation. While this can be viewed as a drawback, the main advantage of generated factor scores is that they are more replicable across studies (Hair et al., 2013).

The reliability of any generated factor score should be examined (Hair et al., 2013) as the final step prior to its use in subsequent analysis. The *reliability* of a scale is estimated as 1 minus the proportion of error variance (R. Kline, 2011). *Cronbach's alpha*, given by

$$\alpha = \left(\frac{K}{K-1}\right)\left(\frac{S_T^2 - \sum_{i=1}^{K} S_i^2}{S_T^2}\right), \tag{13.4}$$

where K is the number of variables, S_i^2 is the variance of each variable, and S_T^2 is the variance of the summated score, is commonly computed to assess the reliability of generated factor scores. It ranges from 0 to 1. A coefficient of 0.7 or greater is usually considered satisfactory; a coefficient of 0.7 would mean that 70% of this scale is reliable or alternatively 30% of the variance is due to error.

13.4 Example in R

In this section, we will estimate an exploratory factor model using R based on a real-life dataset, `workout3`, that is included in the `astatur` package accompanying this book. Our data was collected from members of a training/fitness centre in 2014 in a medium-sized city in Norway. The members were asked, using an ordinal scale (from 1 = *not at all important* to 6 = *very important*), to indicate how important each of the following reasons was for working out:

Var1 - to help manage stress
Var2 - to release tension
Var3 - to mentally relax
Var4 - to have a good body
Var5 - to improve my appearance
Var6 - to look more attractive

Prior to moving further, we need to load the dataset in R as we have done before and deal with the missing values in the variables so that we have complete data to work with. We will filter out all rows that contain missing values and only keep the complete rows/observations and save these in a new object that we call `workout3_comp`. We do this by making use of the `filter()` function from the `dplyr` package:

```
library(dplyr)
workout3_comp <- filter(workout3,
                        complete.cases(workout3))
```

Our aim here is simply to reduce the data, that is to extract a small number of factors that can capture most of the covariance in the matrix that contains the above-mentioned six variables. We achieve this by following the four main steps of the factor analysis process shown in Figure 13.3 previously.

13.4.1 Determining the number of factors

Before we can estimate our factor model, we have to first decide the number of factors to extract. For this purpose, we can make use of the so-called parallel analysis that we explained earlier. We can run the parallel analysis (with scree plot) in R by using the `fa.parallel()` function from the `psych` package (Revelle, 2018). As the first argument to `fa.parallel()`, we write the name of our dataset, specify the factor extraction method (i.e. PA) as `fm="pa"`, and ask for the eigenvalues for the chosen extraction method, namely factor analysis and not PCA by adding the `fa="fa"` argument as shown below. We also add the argument `SMC=TRUE` so that the estimation can start with the SMCs in the diagonal of the initial matrix. We could alternatively use a correlation matrix as input instead of the raw dataset and run the parallel analysis for PCA using the following command: `fa.parallel(workout3_comp, fa="pc")`. As far as the scree plot is concerned, in Figure 13.9, we see that there are two obvious factors before the curve levels off. The results of the parallel analysis are also contained in this graph: when we look at Figure 13.9, we see that the parallel analysis line (i.e. dashed line) crosses the factor analysis line (i.e. bold line) before reaching the third factor. This observation indicates that we should retain the first two factors. The parallel analysis command also produces numerical findings (stored in the object `paranalysis`), which we can access through the command `print(paranalysis)` below:

```
library(psych)
paranalysis <- fa.parallel(workout3_comp,
                           fm="pa", fa="fa", SMC="TRUE")
```

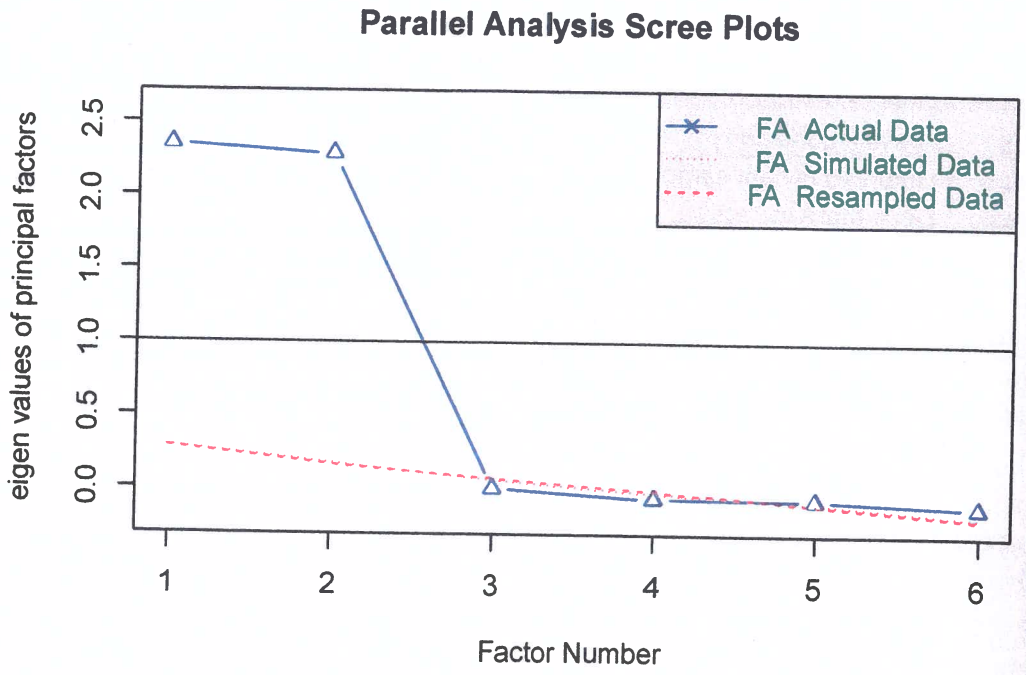


Figure 13.9 Parallel analysis with scree plot

```
## Parallel analysis suggests that the number of factors = 2 and the number of components = NA
```

In the output generated by `print(paranalysis)`, we observe that the largest eigenvalues are 2.34 and 2.28, which both deviate a bit from the corresponding values (2.42 and 2.35) when we estimate our factor model using the main `fa()` function (see Section 13.4.2). The reason for this trivial deviation is that the parallel analysis here is based on a factor analysis without iteration, whereas the main factor analysis is based on iteration. This is the reason why these are referred to in the literature as *principal (axis) factor* and *iterated principal axis factor*, respectively. Based on the numerical findings, we see clear support for a two-factor solution as only the eigenvalues of the first two factors are larger than the average of the eigenvalues of the simulated factors.

```
print(paranalysis)
## Call: fa.parallel(x = workout3_comp, fm = "pa", fa = "fa", SMC = "TRUE")
## Parallel analysis suggests that the number of factors = 2 and the number of components = NA
##
##  Eigen Values of
##
##  eigen values of factors
## [1] 2.34 2.28 0.00 -0.07 -0.08 -0.12
##
##  eigen values of simulated factors
## [1] 0.26 0.15 0.06 -0.03 -0.10 -0.19
##
```

```
##  eigen values of components
## [1] 2.61 2.56 0.32 0.23 0.16 0.12
##
##  eigen values of simulated components
## [1] NA
```

Another criterion to be used for deciding the number of factors is to identify those factors with eigenvalues larger than the average of the SMCs used in the initial correlation matrix that the estimation started with. We can readily obtain these SMC values and take their average with the command below. Here we use the `smc()` function from the `dplyr` package and the `mean()` function from base-R:

```
squaredmc <- smc(workout3_comp)
squaredmc
##      Var1      Var2      Var3      Var4      Var5      Var6
## 0.7020417 0.7899923 0.7098107 0.6124627 0.7871944 0.7465444
mean(squaredmc)
## [1] 0.7246744
```

As we see in the above output, the average SMC is 0.725. When we examine the eigenvalues from the parallel analysis, we observe that only two of these exceed 0.725, a finding that supports our choice to extract two factors.

In addition to the quantitative evidence provided by the above criteria, we can also justify the two-factor solution using theoretical reasoning. Looking at the content of the first three variables (`Var1` to `Var3`), we see that they all are about relaxation, whereas the remaining three variables (`Var4` to `Var6`) revolve around appearance.

13.4.2 Extracting with rotation

The `psych` package in R provides us with the possibility to estimate a factor model based on the two most common extraction methods, namely the PA via the `fa()` function and the PCA through the `principal()` function. In addition, the `psych` package allows us to use several other extraction methods via the `fa()` function. These are *ml*, *minres*, *uls*, *ols*, *wls*, *gls*, *minchi*, *minrank*, *old.min*, and *alpha*. In this chapter, we choose to use PA to estimate our factor model as it is considered to be superior to PCA in most applications.

When it comes to the application of the `fa()` function, we start by writing the name of our dataset or the correlation matrix. Next, we type in the number of factors that we want to extract. Since in the previous section we arrived at two factors, we specify this with `nfactors=2`. As we also want to use PA as the extraction method, we need to declare this using `fm="pa"`. If we do not do this, the factor model will be estimated based on the default method which is *minres*. Finally, we have to indicate if we want to rotate the factor solution. The `fa()` function uses *oblimin* as the default rotation technique. Since we want to use *varimax*, we specify this choice using `rotate="varimax"`. If we would rather like to calculate the unrotated factor solution, we would type `rotate="none"`, instead. The function `fa()` offers several other rotation techniques such as *quartimax*, *equamax*, *promax*, and so on. For a complete overview of these, see the help file of `fa()`.

The rotation helps to polarize the factor loadings even further. The reason why we have chosen *varimax* is that our two factors represent two different phenomena (`relaxation` and `appearance`), and thus we do not assume a strong correlation between the factors. It is incidentally important to remember that `fa()` uses iterative PA for the estimation of our factor model. If we wanted to estimate this factor model using PA with no iteration, then we would have to add the argument `max.iter=0` in the function `fa()`. To be able to see the factor loadings from this solution, we will type in `print(fmodel1$loadings, cutoff=0)` after the estimation. Note that the function `fa()` uses the argument `min.err=0.001`, meaning that the estimation iterates until the change in communality is less than 0.001. This could be a reason why the results from `fa()` may deviate, though trivially, from the same factor models estimated with other statistical software such as Stata, SPSS, and so on.

```
fmodel1 <- fa(workout3_comp,
              nfactors = 2,
              fm="pa",
              rotate = "varimax")
```

Having estimated the factor model and saved the associated results in the object `fmodel1`, we can start asking for the results we want to see bit by bit by using the relevant commands as done below. This bit-by-bit approach is convenient because getting all the estimation results from the factor analysis can be too much information to digest at once. If we wanted to get the complete results from the factor analysis, then we could type in `print(fmodel1)`. When we ask for a specific segment of the results/output from the estimated model, we first write the name of the object the estimation is stored in (i.e. `fmodel1`) and then continue with the `$` sign, and finally type in the name of the object that specific output is stored in (e.g. `n.obs`). To be able to see all the objects that contain different bits of information from the factor estimation, we can apply the `attributes()` function from base-R as `attributes(fmodel1)`.

We can start interpreting the results by first asking for information about the number of observations used for estimating the factor model:

```
print(fmodel1$n.obs)
## [1] 194
```

Here we see that there are in total 194 observations used in the factor model estimation. Next, we can ask for the overview of the factor loadings as well as the eigenvalues for the factor solution:

```
print(fmodel1$loadings, digits=4, cutoff=0)
##
## Loadings:
##       PA1      PA2
## Var1  0.8662 -0.0287
## Var2  0.9522 -0.0480
## Var3  0.8689  0.0465
## Var4  0.0575  0.7990
## Var5  0.0093  0.9550
## Var6 -0.0382  0.8943
```

```
##
##                        PA1    PA2
## SS loadings          2.4168 2.3554
## Proportion Var       0.4028 0.3926
## Cumulative Var       0.4028 0.7954
```

As we see in the output above, the first three variables (`Var1` to `Var3`) load strongly on factor 1 and weakly on factor 2, whereas the remaining variables (`Var4` to `Var6`) load strongly on factor 2 and weakly on factor 1. If we further want to display the relationship between the variables and factors geometrically, we can use the function `fa.plot()` from the `psych` package.

As we can see, Figure 13.10 confirms the interpretation of the factor loadings above. Incidentally, the rotation does not change the total variance explained. The total variance will be equal to the sum of the variable (item) communalities. Instead of SMCs (that usually do not provide best estimates), we use communalities that we get after the estimation to be able to compute the total variance. We can obtain the variables (item) communalities and take their sum with the command below:

```
comm <- fmodel1$communality
comm
##      Var1      Var2      Var3      Var4      Var5      Var6
## 0.7511719 0.9088973 0.7572130 0.6417337 0.9120803 0.8011432
sum(comm)
## [1] 4.772239
```

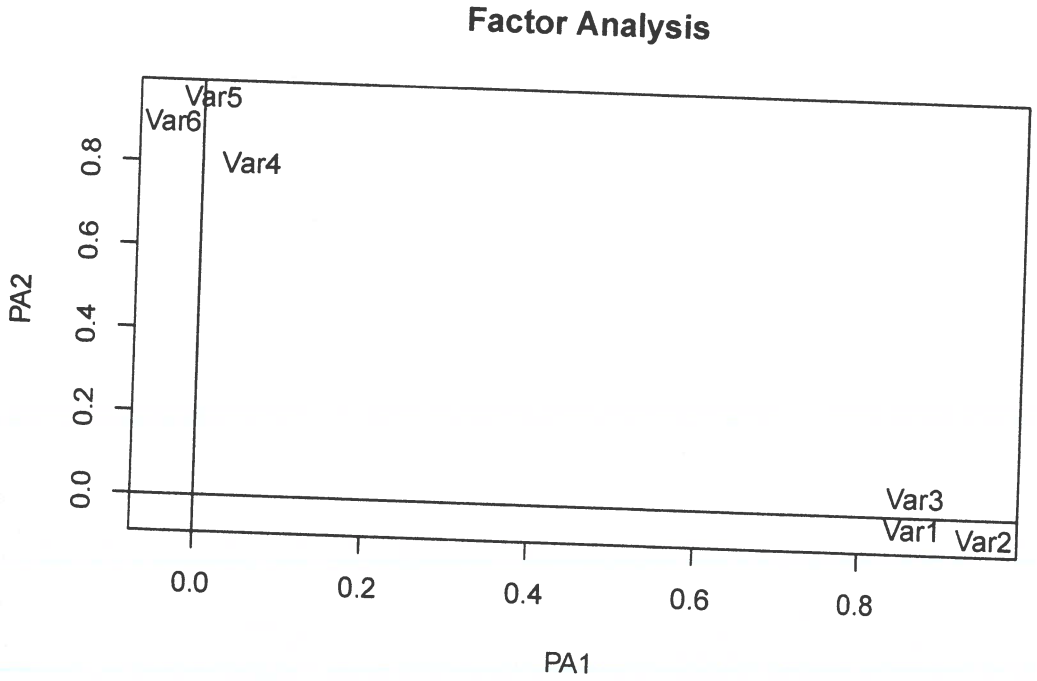


Figure 13.10 Plot of factor loadings after rotation

Here we observe that the sum of the variable (item) communalities is 4.77. Remember that if we had used PCA, the total variance would have been 6 instead. To be able to find out what share of the total variance each of the two factors explains, we divide the eigenvalues from the factor solution by the total variance. As such, we can see that factor 1 (2.42/4.77) and factor 2 (2.36/4.77) explain about 50% of the variance each.

As far as the factor loading matrix above (generated by `fmodel1$loadings`) is concerned, we suggest that one goes about it first vertically and then horizontally. When interpreting the matrix vertically, we locate the correlations^2 between a factor and all the observed variables. For instance, the loading of `Var1` on factor 1 is 0.8662. This means that nearly 75% (0.8662^2) of the variance of `Var1` is explained by factor 1. We interpret the remaining loadings in the same manner. When we interpret the matrix horizontally, we obtain item communalities. For instance, we see that the loadings of `Var3` on factor 1 and factor 2 are 0.8689 and 0.0465, respectively. Squaring and adding these two values together would show us the proportion (about 75%) of the total variance of `Var3` that is explained by factor 1 and factor 2 together. What is left (about 25%) represents the unique variance (i.e. unexplained variance). Although we can compute the item communalities (h^2) and unique variance (u^2) from the factor loadings manually, we can get these computed for us in a much quicker way with the command below:

```
cbind(h2=fmodel1$communality, u2=fmodel1$uniquenesses)
##              h2         u2
## Var1 0.7511719 0.24882811
## Var2 0.9088973 0.09110270
## Var3 0.7572130 0.24278697
## Var4 0.6417337 0.35826630
## Var5 0.9120803 0.08791973
## Var6 0.8011432 0.19885679
```

Since all the loadings are clearly above our threshold of 0.4, we would go along with the current factor solution. There are no variables loading equally strongly on both of the factors either. The final step of the factor analysis process is labelling the factors that we decide to retain. As we mentioned earlier, having studied the content of the items and loadings, we choose to label factor 1 as `relaxation` and factor 2 as `appearance`.

If the purpose of our factor analysis were to examine only the factor structure (which variables can be underlined by which factors), we could stop the process here. However, we know that many social science studies use factors in subsequent analysis. To go down this path, we first need a metric for the two hypothetical constructs. As you know, we can choose between the *estimated* factor score and the *generated* factor score. Since the loadings are clearly polarized, we choose to generate factor scores. This can be done by taking either the sum or average of the variables expressing each factor. We will choose to take the average to keep the factor metric on the same scale (1–6) as the original observed variables. If we want to get the *estimated* factor scores (instead of the *generated* ones), we will just type in `fmodel1$scores`. This applies when we use `fa()` and `principal()` functions, both of which use the regression as the default method for computing factor scores. For this purpose, we will use the `scoreItems()` function from the `psych` package below. This function takes a description of which variables belong together. We specify this description as variable `itemlist`:

```
itemlist <- list(relaxation=c("Var1","Var2","Var3"),
                 appearance=c("Var4","Var5","Var6"))
summateds <- scoreItems(itemlist, workout3_comp,
                        min=1, max=6, totals = FALSE)
factordata <- as.data.frame(summateds$scores)
```

We can now add these two newly generated factor scores to our existing main dataset, `workout3_comp`. We achieve this with the help of the `bind_cols()` function from the `dplyr` package that we discussed in Chapter 4:

```
workout3_comp <- bind_cols(workout3_comp, factordata)
names(workout3_comp)
## [1] "Var1"       "Var2"       "Var3"       "Var4"
## [5] "Var5"       "Var6"       "relaxation" "appearance"
```

Next, we also need to test the reliability of these summated scales based on *Cronbach's alpha* coefficient. We do this by using the `alpha()` function from the `psych` package below. We run this function separately for the first three variables (corresponding to factor 1) and the second three variables (corresponding to factor 2). In other situations, we might have to specify explicitly which of the variables belong to each factor or sort the variables in our data frame to match the sequence of factors.

```
relaxation <- data.frame(workout3_comp[,1:3])
alpha(relaxation)$total$std.alpha
## [1] 0.9234774
appearance <- data.frame(workout3_comp[,4:6])
alpha(appearance)$total$std.alpha
## [1] 0.9124331
```

Here we only output the total coefficient. You can inspect the full output of the reliability analysis by simply looking at, for instance, the output of `alpha(appearance)` directly. As we observe above, both of the *Cronbach's alpha* coefficients are satisfactory in that they are clearly above the suggested minimum level of 0.7.

13.5 Conclusion

EFA is a useful statistical technique that has a wide range of uses in the social sciences. Understanding EFA lays a very good foundation for learning CFA, a special case of structural equation modelling, which is the topic of Chapter 14. We have also learnt that factor analysis is a more subjective statistical technique than traditional techniques such as linear regression as many analytical decisions have to be made and those choices depend on whether or not the obtained solution 'makes sense' to the researcher. This feature of factor analysis thus requires a subjective element of the analyst in an attempt to ascertain the optimal factor solution. In this chapter, we have also presented several useful factor-related functions of the `psych` package in R that in fact offers some additional functions that can be explored by just typing `help(psych)`.

Key terms

Communality The squared multiple correlation of each variable with all the other variables in the correlation matrix
Cronbach's alpha A measure of reliability
Eigenvalue The amount of variance captured/explained by a factor
Eigenvectors Sets of weights that generate factors with the largest possible eigenvalues
Factor extraction A method used to explain the variance in the correlation matrix
Factor loading The correlation (bivariate or partial) between the observed variable and factor
Factor score Value of a factor computed for each observation
Oblique rotation A factor rotation technique allowing for correlation between factors when extracting the factors
Orthogonal rotation A factor rotation technique assuming no correlation between factors when extracting the factors
Parallel analysis A technique used to decide the number of factors to retain based on simulating random datasets
Promax A type of oblique rotation technique
Reliability The consistency between the items or observed variables
Uniqueness The amount of variance not captured/explained by the factor(s)
Varimax A type of orthogonal rotation technique

Questions

1 What is the difference between principal component and principal factor extraction methods?
2 What is the rationale behind rotating factors?
3 How do you determine the number of factors to retain?
4 What is a factor score?
5 Explain what *Cronbach's alpha* is used for and how it is used.

Further reading

Fabrigar, L. R. and Wegener, D. T. (2011). *Exploratory factor analysis*. Oxford University Press, Oxford.
This is a good short introduction to exploratory factor analysis. The authors explain mainly in a non-technical manner the different steps of the factor analysis process with some example factor models estimated at the end of the book.

Hair, J. F., Black, W. C., Babin, B. J., and Anderson, R. E. (2013). *Multivariate data analysis* (7th ed.). Pearson, Harlow.
This comprehensive multivariate statistics book includes a relatively lengthy chapter on factor analysis. The authors explain the salient aspects of factor analysis without mathematical details, leading the reader through an example factor analysis study.

Kline, P. (1994). *An easy guide to factor analysis*. Routledge, New York, NY.
This book first explains principal component analysis with mathematical formulae before covering the typical common factor analysis methods (principal factor analysis, maximum-likelihood factor analysis). The author's approach really does help one understand the logic behind and difference between PCA and factor analysis.

Pett, M. A., Lackey, N. R., and Sullivan, J. J. (2003). *Making sense of factor analysis: The use of factor analysis for instrument development in health care research*. Sage, Thousand Oaks, CA.
Of the four books in this list, this provides the most comprehensive coverage of factor analysis. The book explains not only the factor analysis technique but also the stage before, namely designing an instrument. The authors provide a short but useful introduction to matrices as well. The authors explain the different stages with examples and interpretation of software output.

Examples of functions used in this chapter

psych

```
paranalysis <- fa.parallel(workout3_comp,
     fm="pa", fa="fa", SMC="TRUE")
print(paranalysis)
```

- parallel analysis with scree test for factor analysis

```
paranalysis2 <- fa.parallel(workout3_comp, fa="pc")
print(paranalysis2)
```

- parallel analysis with scree test for PCA

```
smc(workout3_comp)
```

- SMC of variables in a dataset

```
fmodel1 <- fa(workout3_comp, nfactors = 2, fm="pa",
              rotate = "varimax")
```

- principal (*axis*) factor analysis with *varimax* rotation

```
fmodel2 <- principal(workout3_komp, nfactors = 2,
                     rotate = "varimax")
```

- PCA with *varimax* rotation

```
print(fmodel1$loadings, digits=4, cutoff=0)
```

- provides factor loadings and eigenvalues

```
sum(fmodel1$communality)
```

- sum of communalities

```
cbind(h2=fmodel1$communality, u2=fmodel1$uniquenesses)
```

- gives communalities and unique variance

```
fmodel1$scores
```

- provides factor scores

```
itemlist <- list(relaxation=c("Var1","Var2","Var3"),
appearance=c("Var4","Var5","Var6"))
```

- creates two lists, each including three variables

```
summatedsc <- scoreItems(itemlist, workout3_comp,
                         min=1, max=6, totals = FALSE)
```

- takes the mean of the items specified in the item list above

```
factordata <- as.data.frame(summatedsc$scores)
```

- puts the summated scores in a data frame

```
relaxation <- data.frame(workout3_comp[,1:3])
alpha(relaxation)$total$std.alpha
```

- computes Cronbach's alpha of relaxation

Notes

1 This is true for all methods except the maximum-likelihood extraction that manipulates off-diagonal elements rather than values in the diagonals (Tabachnick and Fidell, 2014, p. 688).

2 Note that in orthogonally rotated solutions, loadings are simply correlations.