14 Principal components and factor analysis

This chapter covers

- Principal components analysis
- Exploratory factor analysis
- Other latent variable models

One of the most challenging aspects of multivariate data is the sheer complexity of the information. If you have a dataset with 100 variables, how do you make sense of all the interrelationships present? Even with 20 variables, there are 190 pairwise correlations to consider when you're trying to understand how the individual variables relate to one another. Two related but distinct methodologies for exploring and simplifying complex multivariate data are principal components and exploratory factor analysis.

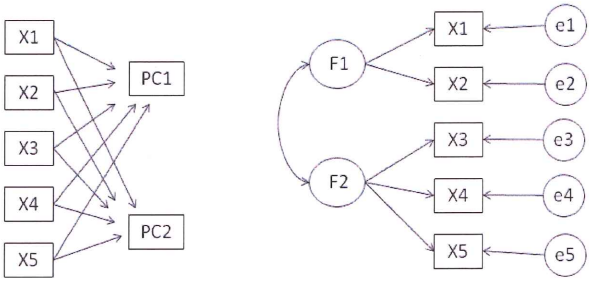
Principal components analysis (PCA) is a data reduction technique that transforms a larger number of correlated variables into a much smaller set of uncorrelated variables called principal components. For example, you might use PCA to transform 30 correlated (and possibly redundant) environmental variables into five uncorrelated composite variables that retain as much information from the original set of variables as possible.

In contrast, exploratory factor analysis (EFA) is a collection of methods designed to uncover the latent structure in a given set of variables. It looks for a smaller set of underlying or latent constructs that can explain the relationships among the observed or manifest variables. For example, the dataset `Harman74.cor` contains the correlations among 24 psychological tests given to 145 seventh- and eighth-grade children. If you apply EFA to this data, the results suggest that the 276 test intercorrelations can be explained by the children's abilities on four underlying factors (verbal ability, processing speed, deduction, and memory).

The differences between the PCA and EFA models can be seen in figure 14.1. Principal components (PC1 and PC2) are linear combinations of the observed variables (X1 to X5). The weights used to form the linear composites are chosen to maximize the variance each principal component accounts for, while keeping the components uncorrelated.

In contrast, factors (F1 and F2) are assumed to underlie or "cause" the observed variables, rather than being linear combinations of them. The errors (e1 to e5) represent the variance in the observed variables unexplained by the factors. The circles indicate that the factors and errors aren't directly observable but are inferred from the correlations among the variables. In this example, the curved arrow between the factors indicates that they're correlated. Correlated factors are common, but not required, in the EFA model.

The methods described in this chapter require large samples to derive stable solutions. What constitutes an adequate sample size is somewhat complicated. Until recently, analysts used rules of thumb like "factor analysis requires 5–10 times as many subjects as variables." Recent studies suggest that the required sample size depends on the number of factors, the number of variables associated with each factor, and how well the set of factors explains the variance in the variables (Bandalos and Boehm-Kaufman, 2009). I'll go out on a limb and say that if you have several hundred observations, you're probably safe. In this chapter, we'll look at artificially small problems in order to keep the output (and page count) manageable.



(a) Principal Components Model (b) Factor Analysis Model

Figure 14.1 Comparing principal components and factor analysis models. The diagrams show the observed variables (X1 to X5), the principal components (PC1, PC2), factors (F1, F2), and errors (e1 to e5).

We'll start by reviewing the functions in R that can be used to perform PCA or EFA and give a brief overview of the steps involved. Then we'll work carefully through two PCA examples, followed by an extended EFA example. A brief overview of other packages in R that can be used for fitting latent variable models is provided at the end of the chapter. This discussion includes packages for confirmatory factor analysis, structural equation modeling, correspondence analysis, and latent class analysis.

14.1 Principal components and factor analysis in R

In the base installation of R, the functions for PCA and EFA are `principomp()` and `factanal()`, respectively. In this chapter, we'll focus on functions provided in the `psych` package. They offer many more useful options than their base counterparts. Additionally, the results are reported in a metric that will be more familiar to social scientists and more likely to match the output provided by corresponding programs in other statistical packages such as SAS and SPSS.

The `psych` package functions that are most relevant here are listed in table 14.1. Be sure to install the package before trying the examples in this chapter.

Table 14.1 Useful factor analytic functions in the `psych` package

Function	Description
`principal()`	Principal components analysis with optional rotation
`fa()`	Factor analysis by principal axis, minimum residual, weighted least squares, or maximum likelihood
`fa.parallel()`	Scree plots with parallel analyses
`factor.plot()`	Plot the results of a factor or principal components analysis
`fa.diagram()`	Graph factor or principal components loading matrices
`scree()`	Scree plot for factor and principal components analysis

EFA (and to a lesser degree PCA) are often confusing to new users. The reason is that they describe a wide range of approaches, and each approach requires several steps (and decisions) to achieve a final result. The most common steps are as follows:

1. *Prepare the data.* Both PCA and EFA derive their solutions from the correlations among the observed variables. Users can input either the raw data matrix or the correlation matrix to the `principal()` and `fa()` functions. If raw data is input, the correlation matrix will automatically be calculated. Be sure to screen the data for missing values before proceeding.
2. *Select a factor model.* Decide whether PCA (data reduction) or EFA (uncovering latent structure) is a better fit for your research goals. If you select an EFA approach, you'll also need to choose a specific factoring method (for example, maximum likelihood).

3. *Decide how many components/factors to extract.*
4. *Extract the components/factors.*
5. *Rotate the components/factors.*
6. *Interpret the results.*
7. *Compute component or factor scores.*

In the remainder of this chapter, we'll carefully consider each of the steps, starting with PCA. At the end of the chapter, you'll find a detailed flow chart of the possible steps in PCA/EFA (figure 14.7). The chart will make more sense once you've read through the intervening material.

14.2 Principal components

The goal of PCA is to replace a large number of correlated variables with a smaller number of uncorrelated variables while capturing as much information in the original variables as possible. These derived variables, called principal components, are linear combinations of the observed variables. Specifically, the first principal component

$$PC_1 = a_1X_1 + a_2X_2 + \ldots + a_kX_k$$

is the weighted combination of the k observed variables that accounts for the most variance in the original set of variables. The second principal component is the linear combination that accounts for the most variance in the original variables, under the constraint that it's *orthogonal* (uncorrelated) to the first principal component. Each subsequent component maximizes the variance accounted for, while at the same time remaining uncorrelated with all previous components. Theoretically, you can extract as many principal components as there are variables. But from a practical viewpoint, you hope that you can approximate the full set of variables with a much smaller set of components. Let's look at a simple example.

The dataset `USJudgeRatings` contains lawyers' ratings of state judges in the US Superior Court. The data frame contains 43 observations on 12 numeric variables. The variables are listed in table 14.2.

Table 14.2 Variables in the `USJudgeRatings` dataset

Variable	Description	Variable	Description
CONT	Number of contacts of lawyer with judge	PREP	Preparation for trial
INTG	Judicial integrity	FAMI	Familiarity with law
DMNR	Demeanor	ORAL	Sound oral rulings
DILG	Diligence	WRIT	Sound written rulings
CFMG	Case flow managing	PHYS	Physical ability
DECI	Prompt decisions	RTEN	Worthy of retention

From a practical point of view, can you summarize the 11 evaluative ratings (INTG to RTEN) with a smaller number of composite variables? If so, how many will you need and how will they be defined? Because our goal is to simplify the data, we'll approach this problem using PCA. The data are in raw score format and there are no missing values. Therefore, your next decision is deciding how many principal components you'll need.

14.2.1 Selecting the number of components to extract

Several criteria are available for deciding how many components to retain in a PCA. They include:

- Basing the number of components on prior experience and theory
- Selecting the number of components needed to account for some threshold cumulative amount of variance in the variables (for example, 80 percent)
- Selecting the number of components to retain by examining the eigenvalues of the k x k correlation matrix among the variables

The most common approach is based on the eigenvalues. Each component is associated with an eigenvalue of the correlation matrix. The first PC is associated with the largest eigenvalue, the second PC with the second-largest eigenvalue, and so on. The Kaiser–Harris criterion suggests retaining components with eigenvalues greater than 1. Components with eigenvalues less than 1 explain less variance than contained in a single variable. In the Cattell Scree test, the eigenvalues are plotted against their component numbers. Such plots will typically demonstrate a bend or elbow, and the components above this sharp break are retained. Finally, you can run simulations, extracting eigenvalues from random data matrices of the same size as the original matrix. If an eigenvalue based on real data is larger than the average corresponding eigenvalues from a set of random data matrices, that component is retained. The approach is called *parallel analysis* (see Hayton, Allen, and Scarpello, 2004 for more details).

You can assess all three eigenvalue criteria at the same time via the `fa.parallel()` function. For the 11 ratings (dropping the `CONT` variable), the necessary code is as follows:

```
library(psych)
fa.parallel(USJudgeRatings[,-1], fa="PC", n.iter=100,
            show.legend=FALSE, main="Scree plot with parallel analysis")
```

This code produces the graph shown in figure 14.2. The plot displays the scree test based on the observed eigenvalues (as straight-line segments and x's), the mean eigenvalues derived from 100 random data matrices (as dashed lines), and the eigenvalues greater than 1 criteria (as a horizontal line at y=1).

All three criteria suggest that a single component is appropriate for summarizing this dataset. Your next step is to extract the principal component using the `principal()` function.

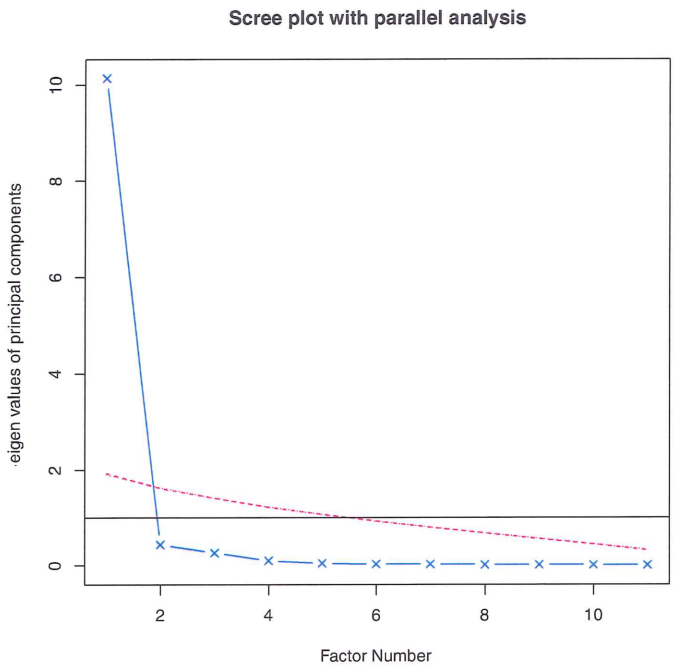


Figure 14.2 Assessing the number of principal components to retain for the US Judge Rating example. A scree plot (the line with x's), eigenvalues greater than 1 criteria (horizontal line), and parallel analysis with 100 simulations (dashed line) suggest retaining a single component.

14.2.2 Extracting principal components

As indicated earlier, the `principal()` function will perform a principal components analysis starting with either a raw data matrix or a correlation matrix. The format is

```
principal(r, nfactors=, rotate=, scores=)
```

where

- `r` is a correlation matrix or a raw data matrix
- `nfactors` specifies the number of principal components to extract (1 by default)
- `rotate` indicates the rotation to be applied (varimax by default; see section 14.2.3)
- `scores` specifies whether or not to calculate principal component scores (false by default)

To extract the first principal component, you can use the code in the following listing.

Listing 14.1 Principal components analysis of US Judge Ratings

```
> library(psych)
> pc <- principal(USJudgeRatings[,-1], nfactors=1)
> pc
```

```
Principal Components Analysis
Call: principal(r = USJudgeRatings[, -1], nfactors=1)
Standardized loadings based upon correlation matrix
      PC1   h2    u2
INTG 0.92 0.84 0.157
DMNR 0.91 0.83 0.166
DILG 0.97 0.94 0.061
CFMG 0.96 0.93 0.072
DECI 0.96 0.92 0.076
PREP 0.98 0.97 0.030
FAMI 0.98 0.95 0.047
ORAL 1.00 0.99 0.009
WRIT 0.99 0.98 0.020
PHYS 0.89 0.80 0.201
RTEN 0.99 0.97 0.028

                 PC1
SS loadings    10.13
Proportion Var  0.92
[… additional output omitted …]
```

Here, you're inputting the raw data without the `CONT` variable and specifying that one unrotated component should be extracted. (Rotation will be explained in section 14.3.3.) Because PCA is performed on a correlation matrix, the raw data is automatically converted to a correlation matrix before extracting the components.

The column labeled PC1 contains the component *loadings*, which are the correlations of the observed variables with the principal component(s). If you had extracted more than one principal component, there would be columns for PC2, PC3, and so on. Component loadings are used to interpret the meaning of components. You can see that each variable correlates highly with the first component (PC1). It therefore appears to be a general evaluative dimension.

The column labeled `h2` contains the component *communalities*—the amount of variance in each variable explained by the components. The `u2` column contains the component *uniquenesses*, the amount of variance not accounted for by the components (or 1–h2). For example, 80 percent of the variance in physical ability (`PHYS`) ratings is accounted for by the first PC, and 20 percent is not. `PHYS` is the variable least well represented by a one-component solution.

The row labeled `SS loadings` contains the eigenvalues associated with the components. The eigenvalues are the standardized variance associated with a particular component (in this case, the value for the first component is 10.). Finally, the row labeled `Proportion Var` represents the amount of variance accounted for by each component. Here you see that the first principal component accounts for 92 percent of the variance in the 11 variables.

Let's consider a second example, one that results in a solution with more than one principal component. The dataset `Harman23.cor` contains data on 8 body measurements for 305 girls. In this case, the dataset consists of the correlations among the variables rather than the original data (see table 14.3).

Table 14.3 Correlations among body measurements for 305 girls (`Harman23.cor`)

	height	arm span	forearm	lower leg	weight	bitro diameter	chest girth	chest width
height	1.00	0.85	0.80	0.86	0.47	0.40	0.30	0.38
arm span	0.85	1.00	0.88	0.83	0.38	0.33	0.28	0.41
forearm	0.80	0.88	1.00	0.80	0.38	0.32	0.24	0.34
lower.leg	0.86	0.83	0.8	1.00	0.44	0.33	0.33	0.36
weight	0.47	0.38	0.38	0.44	1.00	0.76	0.73	0.63
bitro diameter	0.40	0.33	0.32	0.33	0.76	1.00	0.58	0.58
chest girth	0.30	0.28	0.24	0.33	0.73	0.58	1.00	0.54
chest width	0.38	0.41	0.34	0.36	0.63	0.58	0.54	1.00

Source: Harman, H. H. (1976) *Modern Factor Analysis, Third Edition Revised*, University of Chicago Press, Table 2.3.

Again, you wish to replace the original physical measurements with a smaller number of derived variables. You can determine the number of components to extract using the following code. In this case, you need to identify the correlation matrix (the `cov` component of the `Harman23.cor` object) and specify the sample size (`n.obs`):

```
library(psych)
fa.parallel(Harman23.cor$cov, n.obs=302, fa="pc", n.iter=100,
            show.legend=FALSE, main="Scree plot with parallel analysis")
```

The resulting graph is displayed in figure 14.3.

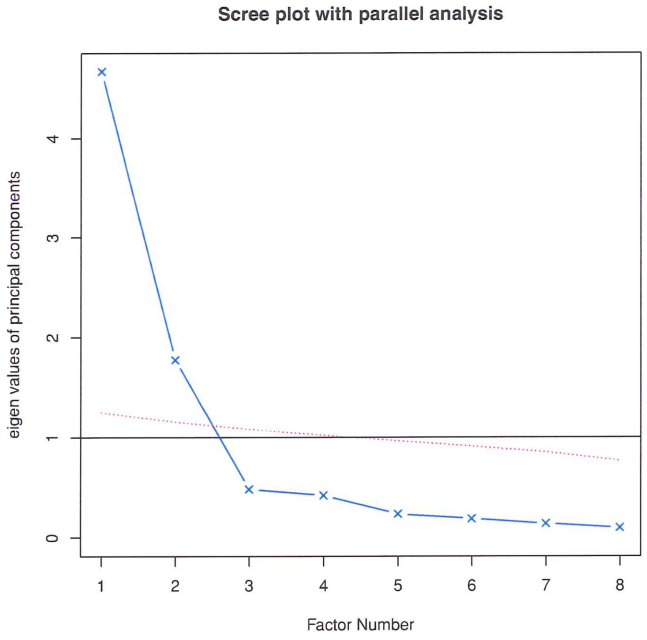


Figure 14.3 Assessing the number of principal components to retain for the Body Measurements example. The scree plot (line with x's), eigenvalues greater than 1 criteria (horizontal line), and parallel analysis with 100 simulations (dashed line) suggest retaining two components.

You can see from the plot that a two-component solution is suggested. As in the first example, the Kaiser–Harris criteria, scree test, and parallel analysis agree. This won't always be the case, and you may need to extract different numbers of components and select the solution that appears most useful. The next listing extracts the first two principal components from the correlation matrix.

Listing 14.2 Principal components analysis of body measurements

```
> library(psych)
> PC <- principal(Harman23.cor$cov, nfactors=2, rotate="none")
> PC

Principal Components Analysis
Call: principal(r = Harman23.cor$cov, nfactors = 2, rotate = "none")
Standardized loadings based upon correlation matrix
                PC1   PC2   h2    u2
height         0.86 -0.37 0.88 0.123
arm.span       0.84 -0.44 0.90 0.097
forearm        0.81 -0.46 0.87 0.128
lower.leg      0.84 -0.40 0.86 0.139
weight         0.76  0.52 0.85 0.150
bitro.diameter 0.67  0.53 0.74 0.261
chest.girth    0.62  0.58 0.72 0.283
chest.width    0.67  0.42 0.62 0.375

                PC1  PC2
SS loadings    4.67 1.77
Proportion Var 0.58 0.22
Cumulative Var 0.58 0.81

[… additional output omitted …]
```

If you examine the `PC1` and `PC2` columns in listing 14.2, you see that the first component accounts for 58 percent of the variance in the physical measurements, while the second component accounts for 22 percent. Together, the two components account for 81 percent of the variance. The two components together account for 88 percent of the variance in the height variable.

Components and factors are interpreted by examining their loadings. The first component correlates positively with each physical measure and appears to be a general size factor. The second component contrasts the first four variables (`height`, `arm.span`, `forearm`, and `lower.leg`), with the second four variables (`weight`, `bitro.diameter`, `chest.girth`, and `chest.width`). It therefore appears to be a length-versus-volume factor. Conceptually, this isn't an easy construct to work with. Whenever two or more components have been extracted, you can rotate the solution to make it more interpretable. This is the topic we'll turn to next.

14.2.3 Rotating principal components

Rotations are a set of mathematical techniques for transforming the component loading matrix into one that's more interpretable. They do this by "purifying" the

components as much as possible. Rotation methods differ with regard to whether the resulting components remain uncorrelated (*orthogonal rotation*) or are allowed to correlate (*oblique rotation*). They also differ in their definition of purifying. The most popular orthogonal rotation is the *varimax* rotation, which attempts to purify the columns of the loading matrix, so that each component is defined by a limited set of variables (that is, each column has a few large loadings and many very small loadings). Applying a varimax rotation to the body measurement data, you get the results provided in the next listing. You'll see an example of an oblique rotation in section 14.4.

Listing 14.3 Principal components analysis with varimax rotation

```
> rc <- principal(Harman23.cor$cov, nfactors=2, rotate="varimax")
> rc

Principal Components Analysis
Call: principal(r = Harman23.cor$cov, nfactors = 2, rotate = "varimax")
Standardized loadings based upon correlation matrix
                RC1  RC2   h2    u2
height         0.90 0.25 0.88 0.123
arm.span       0.93 0.19 0.90 0.097
forearm        0.92 0.16 0.87 0.128
lower.leg      0.90 0.22 0.86 0.139
weight         0.26 0.88 0.85 0.150
bitro.diameter 0.19 0.84 0.74 0.261
chest.girth    0.11 0.84 0.72 0.283
chest.width    0.26 0.75 0.62 0.375

                RC1  RC2
SS loadings    3.52 2.92
Proportion Var 0.44 0.37
Cumulative Var 0.44 0.81

[… additional output omitted …]
```

The column names change from `PC` to `RC` to denote rotated components. Looking at the loadings in column `RC1`, you see that the first component is primarily defined by the first four variables (length variables). The loadings in the column `RC2` indicate that the second component is primarily defined by variables 5 through 8 (volume variables). Note that the two components are still uncorrelated and that together, they still explain the variables equally well. You can see that the rotated solution explains the variables equally well because the variable communalities haven't changed. Additionally, the cumulative variance accounted for by the two-component rotated solution (81 percent) hasn't changed. But the proportion of variance accounted for by each individual component has changed (from 58 percent to 44 percent for component 1 and from 22 percent to 37 percent for component 2). This spreading out of the variance across components is common, and technically you should now call them components rather than principal components (because the variance maximizing properties of individual components has not been retained).

Our ultimate goal is to replace a larger set of correlated variables with a smaller set of derived variables. To do this, you need to obtain scores for each observation on the components.

14.2.4 Obtaining principal components scores

In the US Judge Rating example, you extracted a single principal component from the raw data describing lawyers' ratings on 11 variables. The `principal()` function makes it easy to obtain scores for each participant on this derived variable (see the next listing).

Listing 14.4 Obtaining component scores from raw data

```
> library(psych)
> pc <- principal(USJudgeRatings[,-1], nfactors=1, score=TRUE)
> head(pc$scores)
                      PC1
AARONSON,L.H.   -0.1857981
ALEXANDER,J.M.   0.7469865
ARMENTANO,A.J.   0.0704772
BERDON,R.I.      1.1358765
BRACKEN,J.J.    -2.1586211
BURNS,E.B.       0.7669406
```

The principal component scores are saved in the `scores` element of the object returned by the `principal()` function when the option `scores=TRUE`. If you wanted, you could now get the correlation between the number of contacts occurring between a lawyer and a judge and their evaluation of the judge using

```
> cor(USJudgeRatings$CONT, PC$score)
             PC1
[1,] -0.008815895
```

Apparently, there's no relationship between the lawyer's familiarity and his or her opinions!

When the principal components analysis is based on a correlation matrix and the raw data aren't available, getting principal component scores for each observation is clearly not possible. But you can get the coefficients used to calculate the principal components.

In the body measurement data, you have correlations among body measurements, but you don't have the individual measurements for these 305 girls. You can get the scoring coefficients using the code in the following listing.

Listing 14.5 Obtaining principal component scoring coefficients

```
> library(psych)
> rc <- principal(Harman23.cor$cov, nfactors=2, rotate="varimax")
> round(unclass(rc$weights), 2)
               RC1   RC2
height        0.28 -0.05
arm.span      0.30 -0.08
```

```
forearm          0.30 -0.09
lower.leg        0.28 -0.06
weight          -0.06  0.33
bitro.diameter -0.08  0.32
chest.girth     -0.10  0.34
chest.width     -0.04  0.27
```

The component scores are obtained using the formulas

```
PC1 = 0.28*height + 0.30*arm.span + 0.30*forearm + 0.29*lower.leg -
      0.06*weight - 0.08*bitro.diameter - 0.10*chest.girth -
      0.04*chest.width
```

and

```
PC2 = -0.05*height - 0.08*arm.span - 0.09*forearm - 0.06*lower.leg +
       0.33*weight + 0.32*bitro.diameter + 0.34*chest.girth +
       0.27*chest.width
```

These equations assume that the physical measurements have been standardized (mean=0, sd=1). Note that the weights for PC1 tend to be around 0.3 or 0. The same is true for PC2. As a practical matter, you could simplify your approach further by taking the first composite variable as the mean of the standardized scores for the first four variables. Similarly, you could define the second composite variable as the mean of the standardized scores for the second four variables. This is typically what I'd do in practice.

Little Jiffy conquers the world

There's quite a bit of confusion among data analysts regarding PCA and EFA. One reason for this is historical and can be traced back to a program called Little Jiffy (no kidding). Little Jiffy was one of the most popular early programs for factor analysis, and defaulted to a principal components analysis, extracting components with eigenvalues greater than 1 and rotating them to a varimax solution. The program was so widely used that many social scientists came to think of this defaults as synonymous with EFA. Many later statistical packages also incorporated these defaults in their EFA programs.

As I hope you'll see in the next section, there are important and fundamental differences between PCA and EFA. To learn more about the PCA/EFA confusion, see Hayton, Allen, and Scarpello, 2004.

If your goal is to look for latent underlying variables that explain your observed variables, you can turn to factor analysis. This is the topic of the next section.

14.3 Exploratory factor analysis

The goal of EFA is to explain the correlations among a set of observed variables by uncovering a smaller set of more fundamental unobserved variables underlying the data. These hypothetical, unobserved variables are called *factors*. (Each factor is assumed to explain the variance shared among two or more observed variables, so technically, they are called *common factors*.)

The model can be represented as

$$X_i = a_1F_1 + a_2F_2 + \ldots + a_pF_p + U_i$$

where X_i is the ith observed variable (i = 1...k), F_j are the common factors (j=1...p), and p<k. U_i is the portion of variable X_i unique to that variable (not explained by the common factors). The a_i can be thought of as the degree to which each factor contributes to the composition of an observed variable. If we go back to the `Harman74.cor` example at the beginning of this chapter, we'd say that an individual's scores on each of the 24 observed psychological tests is due to a weighted combination of their ability on four underlying psychological constructs.

Although the PCA and EFA models differ, many of the steps will appear similar. To illustrate the process, we'll apply EFA to the correlations among six psychological tests. One hundred twelve individuals were given six tests, including a nonverbal measure of general intelligence (`general`), a picture-completion test (`picture`), a block design test (`blocks`), a maze test (`maze`), a reading comprehension test (`reading`), and a vocabulary test (`vocab`). Can we explain the participants' scores on these tests with a smaller number of underlying or latent psychological constructs?

The covariance matrix among the variables is provided in the dataset `ability.cov`. You can transform this into a correlation matrix using the `cov2cor()` function. There are no missing data present.

```
> options(digits=2)
> covariances <- ability.cov$cov
> correlations <- cov2cor(covariances)
> correlations
        general picture blocks maze reading vocab
general    1.00    0.47   0.55 0.34    0.58  0.51
picture    0.47    1.00   0.57 0.19    0.26  0.24
blocks     0.55    0.57   1.00 0.45    0.35  0.36
maze       0.34    0.19   0.45 1.00    0.18  0.22
reading    0.58    0.26   0.35 0.18    1.00  0.79
vocab      0.51    0.24   0.36 0.22    0.79  1.00
```

Because you're looking for hypothetical constructs that explain the data, you'll use an EFA approach. As in PCA, the next task is to decide how many factors to extract.

14.3.1 Deciding how many common factors to extract

To decide on the number of factors to extract, turn to the `fa.parallel()` function:

```
> library(psych)
> covariances <- ability.cov$cov
> correlations <- cov2cor(covariances)
> fa.parallel(correlations, n.obs=112, fa="both", n.iter=100,
              main="Scree plots with parallel analysis")
```

The resulting plot is shown in figure 14.4. Notice you've requested that the function display results for both a principal components and common factor approach, so that you can compare them (`fa="both"`).

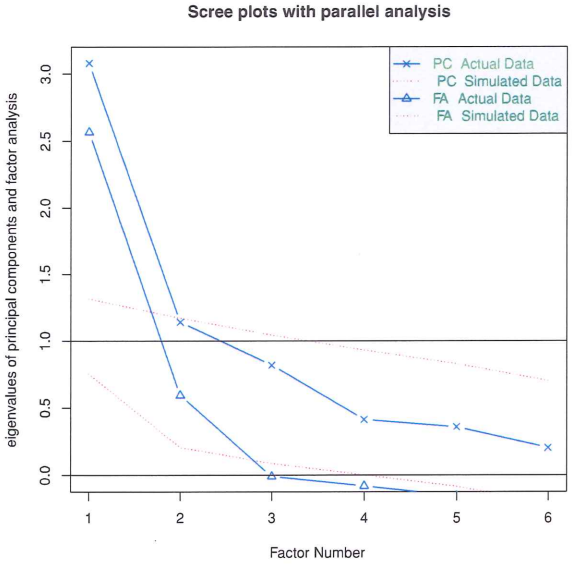


Figure 14.4 Assessing the number of factors to retain for the psychological tests example. Results for both PCA and EFA are present. The PCA results suggest one or two components. The EFA results suggest two factors.

There are several things to notice in this graph. If you'd taken a PCA approach, you might have chosen one component (scree test, parallel analysis) or two components (eigenvalues greater than 1). When in doubt, it's usually a better idea to overfactor than to underfactor. Overfactoring tends to lead to less distortion of the "true" solution.

Looking at the EFA results, a two-factor solution is clearly indicated. The first two eigenvalues (triangles) are above the bend in the scree test and also above the mean eigenvalues based on 100 simulated data matrices. For EFA, the Kaiser–Harris criterion is number of eigenvalues above 0, rather than 1. (Most people don't realize this, so it's a good way to win bets at parties.) In the present case the Kaiser–Harris criteria also suggest two factors.

14.3.2 Extracting common factors

Now that you've decided to extract two factors, you can use the `fa()` function to obtain your solution. The format of the `fa()` function is

```
fa(r, nfactors=, n.obs=, rotate=, scores=, fm=)
```

where

- `r` is a correlation matrix or a raw data matrix
- `nfactors` specifies the number of factors to extract (1 by default)
- `n.obs` is the number of observations (if a correlation matrix is input)
- `rotate` indicates the rotation to be applied (oblimin by default)
- `scores` specifies whether or not to calculate factor scores (false by default)
- `fm` specifies the factoring method (minres by default)

Unlike PCA, there are many methods of extracting common factors. They include maximum likelihood (`ml`), iterated principal axis (`pa`), weighted least square (`wls`),

generalized weighted least squares (`gls`), and minimum residual (`minres`). Statisticians tend to prefer the maximum likelihood approach because of its well-defined statistical model. Sometimes, this approach fails to converge, in which case the iterated principal axis option often works well. To learn more about the different approaches, see Mulaik (2009) and Gorsuch (1983).

For this example, you'll extract the unrotated factors using the iterated principal axis (`fm="pa"`) approach. The results are given in the next listing.

Listing 14.6 Principal axis factoring without rotation

```
> fa <- fa(correlations, nfactors=2, rotate="none", fm="pa")
> fa
Factor Analysis using method =  pa
Call: fa(r = correlations, nfactors = 2, rotate = "none", fm = "pa")
Standardized loadings based upon correlation matrix
          PA1   PA2   h2   u2
general 0.75  0.07 0.57 0.43
picture 0.52  0.32 0.38 0.62
blocks  0.75  0.52 0.83 0.17
maze    0.39  0.22 0.20 0.80
reading 0.81 -0.51 0.91 0.09
vocab   0.73 -0.39 0.69 0.31

                PA1  PA2
SS loadings    2.75 0.83
Proportion Var 0.46 0.14
Cumulative Var 0.46 0.60
[… additional output deleted …]
```

You can see that the two factors account for 60 percent of the variance in the six psychological tests. When you examine the loadings, though, they aren't easy to interpret. Rotating them should help.

14.3.3 Rotating factors

You can rotate the two-factor solution from section 14.3.4 using either an orthogonal rotation or an oblique rotation. Let's try both so you can see how they differ. First try an orthogonal rotation (in the next listing).

Listing 14.7 Factor extraction with orthogonal rotation

```
> fa.varimax <- fa(correlations, nfactors=2, rotate="varimax", fm="pa")
> fa.varimax
Factor Analysis using method =  pa
Call: fa(r = correlations, nfactors = 2, rotate = "varimax", fm = "pa")
Standardized loadings based upon correlation matrix
          PA1  PA2   h2   u2
general 0.49 0.57 0.57 0.43
picture 0.16 0.59 0.38 0.62
blocks  0.18 0.89 0.83 0.17
maze    0.13 0.43 0.20 0.80
reading 0.93 0.20 0.91 0.09
```

```
vocab    0.80 0.23 0.69 0.31

                 PA1  PA2
SS loadings    1.83 1.75
Proportion Var 0.30 0.29
Cumulative Var 0.30 0.60

[… additional output omitted …]
```

Looking at the factor loadings, the factors are certainly easier to interpret. Reading and vocabulary load on the first factor, and picture completion, block design, and mazes loads on the second factor. The general nonverbal intelligence measure loads on both factors. This may indicate a verbal intelligence factor and a nonverbal intelligence factor.

By using an orthogonal rotation, you've artificially forced the two factors to be uncorrelated. What would you find if you allowed the two factors to correlate? You can try an oblique rotation such as *promax* (see the next listing).

Listing 14.8 Factor extraction with oblique rotation

```
> fa.promax <- fa(correlations, nfactors=2, rotate="promax", fm="pa")
> fa.promax
Factor Analysis using method =  pa
Call: fa(r = correlations, nfactors = 2, rotate = "promax", fm = "pa")
Standardized loadings based upon correlation matrix
          PA1   PA2   h2   u2
general  0.36  0.49 0.57 0.43
picture -0.04  0.64 0.38 0.62
blocks  -0.12  0.98 0.83 0.17
maze    -0.01  0.45 0.20 0.80
reading  1.01 -0.11 0.91 0.09
vocab    0.84 -0.02 0.69 0.31

                 PA1  PA2
SS loadings    1.82 1.76
Proportion Var 0.30 0.29
Cumulative Var 0.30 0.60

 With factor correlations of
     PA1  PA2
PA1 1.00 0.57
PA2 0.57 1.00
[… additional output omitted …]
```

Several differences exist between the orthogonal and oblique solutions. In an orthogonal solution, attention focuses on the *factor structure matrix* (the correlations of the variables with the factors). In an oblique solution, there are three matrices to consider: the factor structure matrix, the factor pattern matrix, and the factor intercorrelation matrix.

The *factor pattern matrix* is a matrix of standardized regression coefficients. They give the weights for predicting the variables from the factors. The *factor intercorrelation matrix* gives the correlations among the factors.

In listing 14.8, the values in the `PA1` and `PA2` columns constitute the factor pattern matrix. They're standardized regression coefficients rather than correlations. Examination of the columns of this matrix is still used to name the factors (although there's some controversy here). Again you'd find a verbal and nonverbal factor.

The factor intercorrelation matrix indicates that the correlation between the two factors is 0.57. This is a hefty correlation. If the factor intercorrelations had been low, you might have gone back to an orthogonal solution to keep things simple.

The factor structure matrix (or factor loading matrix) isn't provided. But you can easily calculate it using the formula `F = P*Phi`, where `F` is the factor loading matrix, `P` is the factor pattern matrix, and `Phi` is the factor intercorrelation matrix. A simple function for carrying out the multiplication is as follows:

```
fsm <- function(oblique) {
  if (class(oblique)[2]=="fa" & is.null(oblique$Phi)) {
       warning("Object doesn't look like oblique EFA")
  } else {
       P <- unclass(oblique$loading)
       F <- P %*% oblique$Phi
       colnames(F) <- c("PA1", "PA2")
       return(F)
  }
}
```

Applying this to the example, you get

```
> fsm(fa.promax)
          PA1  PA2
general 0.64 0.69
picture 0.33 0.61
blocks  0.44 0.91
maze    0.25 0.45
reading 0.95 0.47
vocab   0.83 0.46
```

Now you can review the correlations between the variables and the factors. Comparing them to the factor loading matrix in the orthogonal solution, you see that these columns aren't as pure. This is because you've allowed the underlying factors to be correlated. Although the oblique approach is more complicated, it's often a more realistic model of the data.

You can graph an orthogonal or oblique solution using the `factor.plot()` or `fa.diagram()` function. The code

```
factor.plot(fa.promax, labels=rownames(fa.promax$loadings))
```

produces the graph in figure 14.5.

The code

```
fa.diagram(fa.promax, simple=FALSE)
```

produces the diagram in figure 14.6. If you let `simple=TRUE`, only the largest loading per item would be displayed. It shows the largest loadings for each factor, as well as the correlations between the factors. This type of diagram is helpful when there are several factors.

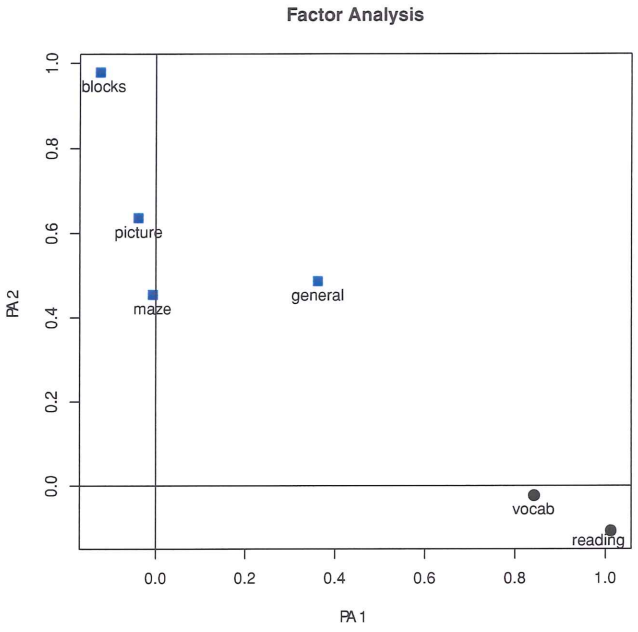


Figure 14.5 Two factor plot for the psychological tests in ability.cov. Vocab and reading load on the first factor (PA1). while blocks, picture, and maze load on the second factor (PA2). The general intelligence test loads on both.

When you're dealing with data in real life, it's unlikely that you'd apply factor analysis to a dataset with so few variables. You've done it here to keep things manageable. If you'd like to test your skills, try factor-analyzing the 24 psychological tests contained in `Harman74.cor`. The code

```
library(psych)
fa.24tests <- fa(Harman74.cor$cov, nfactors=4, rotate="promax")
```

should get you started!

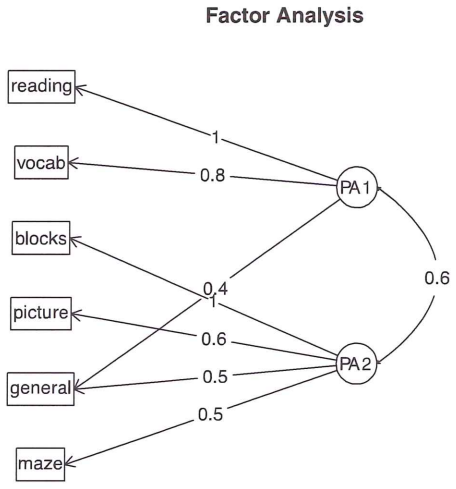


Figure 14.6 Diagram of the oblique two factor solution for the psychological test data in `ability.cov`

14.3.4 Factor scores

Compared with PCA, the goal of EFA is much less likely to be the calculation of factor scores. But these scores are easily obtained from the `fa()` function by including the `score=TRUE` option (when raw data is available). Additionally, the scoring coefficients (standardized regression weights) are available in the `weights` element of the object returned.

For the `ability.cov` dataset, you can obtain the beta weights for calculating the factor score estimates for the two-factor oblique solution using

```
> fa.promax$weights
          [,1]  [,2]
general 0.080 0.210
picture 0.021 0.090
blocks  0.044 0.695
maze    0.027 0.035
reading 0.739 0.044
vocab   0.176 0.039
```

Unlike component scores, which are calculated exactly, factor scores can only be estimated. Several methods exist. The fa() function uses the regression approach. To learn more about factor scores, see DiStefano, Zhu, and Mîndrilă, (2009).

Before moving on, let's briefly review other R packages that are useful for exploratory factor analysis.

14.3.5 Other EFA-related packages

R contains a number of other contributed packages that are useful for conducting factor analyses. The `FactoMineR` package provides methods for PCA and EFA, as well as other latent variable models. It provides many options that we haven't considered here, including the use of both numeric and categorical variables. The `FAiR` package estimates factor analysis models using a genetic algorithm that permits the ability to impose inequality restrictions on model parameters. The `GPArotation` package offers many additional factor rotation methods. Finally, the `nFactors` package offers sophisticated techniques for determining the number of factors underlying data.

14.4 Other latent variable models

EFA is only one of a wide range of latent variable models used in statistics. We'll end this chapter with a brief description of other models that can be fit within R. These include models that test a priori theories, that can handle mixed data types (numeric and categorical), or that are based solely on categorical multiway tables.

In EFA, you allow the data to determine the number of factors to be extracted and their meaning. But you could start with a theory about how many factors underlie a set of variables, how the variables load on those factors, and how the factors correlate with one another. You could then test this theory against a set of collected data. The approach is called confirmatory factor analysis (CFA).

CFA is a subset of a methodology called structural equation modeling (SEM). SEM not only allows you to posit the number and composition of underlying factors but

how these factors impact one another as well. You can think of SEM as a combination of confirmatory factor analyses (for the variables) and regression analyses (for the factors). The resulting output includes statistical tests and fit indices. There are several excellent packages for CFA and SEM in R. They include `sem`, `openMx`, and `lavaan`.

The `ltm` package can be used to fit latent models to the items contained in tests and questionnaires. The methodology is often used to create large scale standardized tests. Examples include the Scholastic Aptitude Test (SAT) and the Graduate Record Exam (GRE).

Latent class models (where the underlying factors are assumed to be categorical rather than continuous) can be fit with the `FlexMix`, `lcmm`, `randomLCA`, and `poLC` packages. The `lcda` package performs latent class discriminant analysis, and the `lsa` package performs latent semantic analysis, a methodology used in natural language processing.

The `ca` package provides functions for simple and multiple correspondence analysis. These methods allow you to explore the structure of categorical variables in two-way and multiway tables, respectively.

Finally, R contains numerous methods for multidimensional scaling (MDS). MDS is designed to detect underlying dimensions that explain the similarities and distances between a set of measured objects (for example, countries). The `cmdscale()` function in the base installation performs a classical MDS, while the `isoMDS()` function in the `MASS` package performs a nonmetric MDS. The `vegan` package also contains functions for classical and nonmetric MDS.

14.5 Summary

In this chapter, we reviewed methods for principal components (PCA) analysis and exploratory factor analysis (EFA). PCA is a useful data reduction method that can replace a large number of correlated variables with a smaller number of uncorrelated variables, simplifying the analyses. EFA contains a broad range of methods for identifying latent or unobserved constructs (factors) that may underlie a set of observed or manifest variables.

Whereas the goal of PCA is typically to summarize the data and reduce its dimensionality, EFA can be used as a hypothesis generating tool, useful when you're trying to understand the relationships between a large number of variables. It's often used in the social sciences for theory development.

Although there are many superficial similarities between the two approaches, important differences exist as well. In this chapter, we considered the models underlying each, methods for selecting the number of components/factors to extract, methods for extracting components/factors and rotating (transforming) them to enhance interpretability, and techniques for obtaining component or factor scores. The steps in a PCA or EFA are summarized in figure 14.7. We ended the chapter with a brief discussion of other latent variable methods available in R.

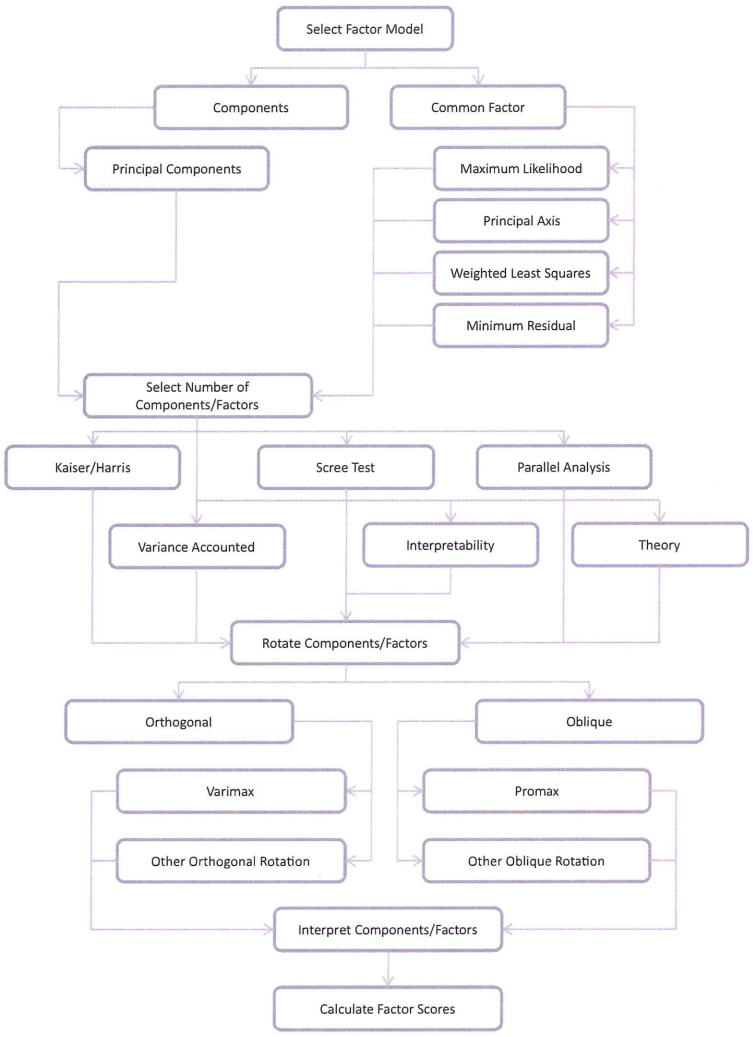


Figure 14.7 A principal components/exploratory factor analysis decision chart

Because PCA and EFA are based on correlation matrices, it's important that any missing data be eliminated before proceeding with the analyses. Section 4.5 briefly mentioned simple methods for dealing with missing data. In the next chapter, we'll consider more sophisticated methods for both understanding and handling missing values.