Chapter 6
Constructs, Components, and Factor models

Parsimony of description has been a goal of science since at least the famous dictum commonly attributed to William of Ockham to not multiply entities beyond necessity[1]. The goal for parsimony is seen in psychometrics as an attempt either to describe (components) or to explain (factors) the relationships between many observed variables in terms of a more limited set of components or latent factors.

The typical data matrix represents multiple items or scales usually thought to reflect fewer underlying constructs[2]. At the most simple, a set of items can be be thought of representing random samples from one underlying domain or perhaps a small set of domains. The question for the psychometrician is how many domains are represented and how well does each item represent the domains. Solutions to this problem are examples of *factor analysis* (*FA*), *principal components analysis* (*PCA*), and *cluster analysis* (*CA*). All of these procedures aim to reduce the complexity of the observed data. In the case of FA, the goal is to identify fewer underlying constructs to explain the observed data. In the case of PCA, the goal can be mere data reduction, but the interpretation of components is frequently done in terms similar to those used when describing the latent variables estimated by FA. Cluster analytic techniques, although usually used to partition the subject space rather than the variable space, can also be used to group variables to reduce the complexity of the data by forming fewer and more homogeneous sets of tests or items.

At the data level the data reduction problem may be solved as a *Singular Value Decomposition* of the original matrix, although the more typical solution is to find either the *principal components* or *factors* of the covariance or correlation matrices. Given the pattern of regression weights from the variables to the components or from the factors to the variables, it is then possible to find (for components) individual *component* or *cluster scores* or estimate (for factors) *factor scores*.

Consider the matrix $\mathbf{X}$ of n deviation scores for N subjects, where each element, x_{ij}, represents the responses of the i^{th} individual to the j^{th} item or test. For simplicity, let the x_{ij} scores in each column be deviations from the mean for that column (i.e., they are column centered, perhaps by using `scale`). Let the number of variables be n. Then the covariance matrix, **Cov**, is

[1] Although probably neither original with Ockham nor directly stated by him (Thorburn, 1918), Ockham's razor remains a fundamental principal of science.

[2] Cattell (1978) as well as MacCallum et al. (2007) argue that the data are the result of many more factors than observed variables, but are willing to estimate the major underlying factors.

$$\mathbf{Cov} = N^{-1}\mathbf{X}\mathbf{X}'$$

and the standard deviations are

$$\mathbf{sd} = \sqrt{diag(\mathbf{Cov})}.$$

Let the matrix $\mathbf{I_{sd}}$ be a diagonal matrix with elements $= \frac{1}{sd_i}$, then the correlation matrix $\mathbf{R}$ is

$$\mathbf{R} = \mathbf{I_{sd}}\mathbf{Cov}\mathbf{I_{sd}}.$$

The problem is how to approximate the matrix, $\mathbf{R}$ of rank n, with a matrix of lower rank? The solution to this problem may be seen if we think about how to create a model matrix to approximate $\mathbf{R}$.

Consider the correlation matrix $\mathbf{R}$ formed as the matrix product of a vector $\mathbf{f}$ (Table 6.1)[3]: By observation, except for the diagonal, $\mathbf{R}$ seems to be a multiplication table with the first

Table 6.1 Creating a correlation matrix from a factor model. In this case, the factor model is a single vector $\mathbf{f}$ and the correlation matrix is created as the product of $\mathbf{ff}'$ with the additional constraint that the diagonal of the resulting matrix is set to 1.

```
> f <- seq(.9,.4,-.1)  #the model
> f
[1] 0.9 0.8 0.7 0.6 0.5 0.4

> R <- f %*% t(f)  #create the correlation matrix
> diag(R) <- 1
> rownames(R) <- colnames(R) <- paste("V",seq(1:6),sep="")
> R
```

```
     V1   V2   V3   V4   V5   V6
V1 1.00 0.72 0.63 0.54 0.45 0.36
V2 0.72 1.00 0.56 0.48 0.40 0.32
V3 0.63 0.56 1.00 0.42 0.35 0.28
V4 0.54 0.48 0.42 1.00 0.30 0.24
V5 0.45 0.40 0.35 0.30 1.00 0.20
V6 0.36 0.32 0.28 0.24 0.20 1.00
```

column representing the .9s, the second column the .8s, etc. Is it possible to represent this matrix in a more parsimonious way? That is, can we determine the vector $\mathbf{f}$ that generated the correlation matrix. (This is sometimes seen as the problem of unscrambling eggs.) There are two broad answers to this question. The first is a model that approximates the correlation matrix in terms of the product of components where each component is a weighted linear sum of the variables, the second model is also an approximation of the correlation matrix by the product of two factors, but the factors in this are seen as causes rather than as consequences of the variables.

That is

$$\mathbf{R} \approx \mathbf{C}\mathbf{C}' \tag{6.1}$$

[3] Although the following discussion will be done in terms of correlation matrices, goodness of fit tests are more typically done on covariance matrices. It is somewhat simpler to do the discussion in terms of correlations.

where, if n is the number of variables in $\mathbf{R}$, then the i^{th} component, C_i, is a linear sum of the variables:

$$C_i = \sum_{j=1}^{n} w_{ij} x_j. \tag{6.2}$$

The factor model appears to be very similar, but with the addition of a diagonal matrix of uniqueness ($\mathbf{U}^2$), and

$$\mathbf{R} \approx \mathbf{F}\mathbf{F}' + \mathbf{U}^2 \tag{6.3}$$

with the variables described as weighted linear sums of the (unknown) factors:

$$x_i \approx \sum_{j=1}^{n} w_{ij} F_j. \tag{6.4}$$

The weights in Equation 6.4 although similar, are not the same as the those in Equation 6.2.

One way to think of the difference between the two models is in terms of *path diagrams*. For the component model, the component is the linear sum of the known variables. This is shown by the arrows from the variables to the component in the left hand panel of Figure 6.1. The factor model, on the other hand, represents each variable as the linear sum of two parts, the common factor and a unique factor for each variable. These effects are shown as arrows from the factor to the variables and curved arrows from the variables to themselves. (The notation represents observed variables with square boxes, unobservable or latent variables with circles or ellipses. The coefficients are the weights to be found.) To solve for the components is a straightforward exercise in linear algebra, to solve the factors is a bit more complicated.

6.1 Principal Components: an observed variable model

Principal components is a method that finds the *basis space* of a particular correlation or covariance matrix. That is, it is a way of finding a matrix of orthogonal vectors that can represent the original matrix. This may be done by finding the *eigenvectors* and *eigenvalues* of the original matrix. The *eigenvectors* are also known as the *characteristic roots* of the matrix, and the *eigenvalues* are simply scalings of the roots.

6.1.1 Eigenvalues and Eigenvectors

As reviewed in Appendix E given a *nxn* matrix $\mathbf{R}$, each eigenvector, $\mathbf{x_i}$, solves the equation

$$\mathbf{x_i R} = \lambda_i \mathbf{x_i}$$

and the set of n eigenvectors are solutions to the equation

$$\mathbf{RX} = \mathbf{X}\lambda$$

where $\mathbf{X}$ is a matrix of orthogonal eigenvectors and λ is a diagonal matrix of the the eigenvalues, λ_i. Then

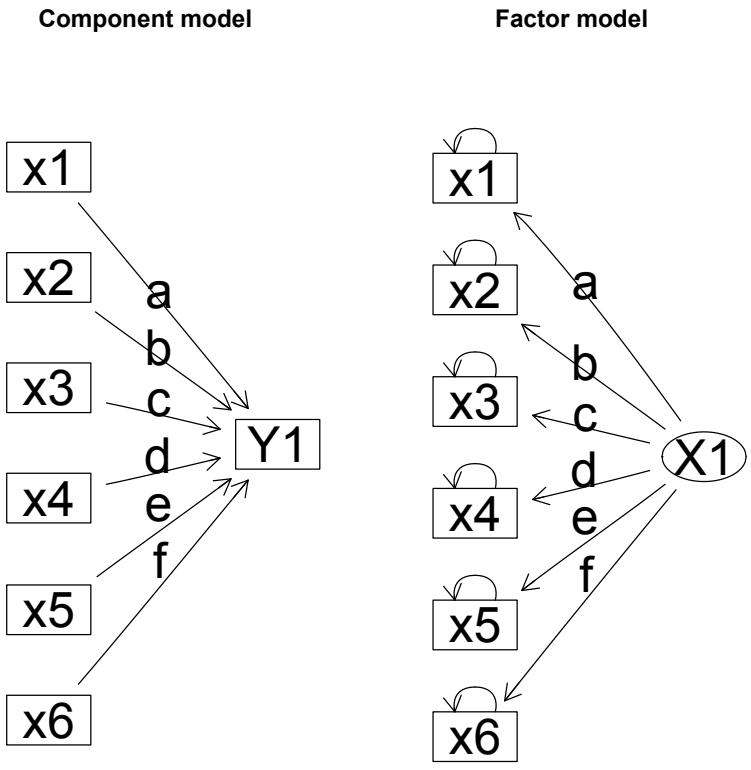

Fig. 6.1 Components are linear sums of variables and do not necessarily say anything about the correlations between the variables. Factors are latent variables thought to explain the correlations or covariances between observed variables. The fundamental factor equation expresses the correlations between variables in terms of the factor loadings. The variance of each variable is made up of two unobservable parts, that which is common to the variables (the factor) and that which is unique to each variable. These unique components are shown as curved arrows back to the variable. Observed variables are represented by boxes, unobserved variables by ellipses. Graph created using `structure.graph`.

$$\mathbf{x_i R} - \lambda_i \mathbf{XI} = 0 <=> \mathbf{x_i}(\mathbf{R} - \lambda_i \mathbf{I}) = 0$$

Finding the eigenvectors and eigenvalues is computationally tedious, but may be done using the `eigen` function which uses a **QR** *decomposition* of the matrix. That the vectors making up $\mathbf{X}$ are orthogonal means that

$$\mathbf{X}\mathbf{X}' = \mathbf{I}$$

and that

$$\mathbf{R} = \mathbf{X}\lambda\mathbf{X}'.$$

That is, it is possible to recreate the correlation matrix $\mathbf{R}$ in terms of an orthogonal set of vectors (the *eigenvectors*) scaled by their associated *eigenvalues*. Both the *eigenvectors* and *eigenvalues* are found by using the `eigen` function.

Principal components are simply the eigenvectors scaled by the square root of the eigenvalues:

$$\mathbf{C} = \mathbf{X}\sqrt{\lambda}$$

and thus $\mathbf{R} = \mathbf{C}\mathbf{C}'$.

Table 6.2 Eigenvalue decomposition of a matrix produces a vector of eigenvalues and a matrix of eigenvectors. The product of the eigenvector with its transpose is the identity matrix. Defining a component as the eigenvector matrix scaled by the squareroot of the eigenvalues leads to recreating the matrix by the product of the component matrix times its transpose.

```
>  e <- eigen(R) #eigenvalue decomposition
>  print(e,digits=2) #show the solution

$values
[1] 3.16 0.82 0.72 0.59 0.44 0.26
$vectors
      [,1]   [,2]   [,3]  [,4]   [,5]   [,6]
[1,] -0.50  0.061  0.092  0.14  0.238  0.816
[2,] -0.47  0.074  0.121  0.21  0.657 -0.533
[3,] -0.43  0.096  0.182  0.53 -0.675 -0.184
[4,] -0.39  0.142  0.414 -0.78 -0.201 -0.104
[5,] -0.34  0.299 -0.860 -0.20 -0.108 -0.067
[6,] -0.28 -0.934 -0.178 -0.10 -0.067 -0.045

round(e$vectors %*% t(e$vectors),2) #show that  the eigenvectors are orthogonal

     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    0    0    0    0    0
[2,]    0    1    0    0    0    0
[3,]    0    0    1    0    0    0
[4,]    0    0    0    1    0    0
[5,]    0    0    0    0    1    0
[6,]    0    0    0    0    0    1

> C <-  e$vectors %*% diag(sqrt(e$values))  #components
> C %*% t(C)   #reproduce the correlation matrix

    [,1] [,2] [,3] [,4] [,5] [,6]
[1,] 1.00 0.72 0.63 0.54 0.45 0.36
[2,] 0.72 1.00 0.56 0.48 0.40 0.32
[3,] 0.63 0.56 1.00 0.42 0.35 0.28
[4,] 0.54 0.48 0.42 1.00 0.30 0.24
[5,] 0.45 0.40 0.35 0.30 1.00 0.20
[6,] 0.36 0.32 0.28 0.24 0.20 1.00
```

6.1.2 Principal components

Typically we do not want to exactly reproduce the original $n * n$ correlation matrix, for there is no gain in parsimony (the rank of the **C** matrix is the same as the rank of the original **R** matrix) but rather want to approximate it with a matrix of lower rank ($k < n$). This may be

done by using just the first k principal components. This requires selecting and rescaling the first k components returned by the functions `princomp` and `prcomp` (Anderson, 1963). Alternatively, the `principal` function will provide the first k components scaled appropriately.

Consider just the first principal component of the matrix **R** (Table 6.3). The *loadings* matrix shows the correlations of each variable with the component. The *uniquenesses*, a concept from factor analysis, reflect the variance not explained for each variable. As is seen in Table 6.3, just one component does not reproduce the matrix very well, for it overestimates the correlations and underestimates the elements on the diagonal. The components solution, in attempting to account for the entire matrix, underestimates the importance of the major variables, and overestimates the importance of the least important variables. This is due to the influence of the diagonal elements of the matrix which are also being fitted. This is most clearly seen by examining the residual matrix of the difference between **R** and the model of **R** which is the product of the first principal component with its transpose. Increasing the number of components used will provide a progressively better approximation to the original **R** matrix, but at a cost of a reduction in parsimony.

If the goal is simple and parsimonious description of a correlation or covariance matrix, the first k principal components will do a better job than any other k-dimensional solution.

6.2 Exploratory Factor Analysis: a latent variable model

Originally developed by Spearman (1904a) for the case of one common factor, and then later generalized by Thurstone (1947) and others to the case of multiple factors, factor analysis is probably the most frequently used and sometimes the most controversial psychometric procedure. The factor model (Equation 6.3), although seemingly very similar to the components model, is in fact very different. For rather than having components as linear sums of variables, in the factor model the variables are themselves linear sums of the unknown factors. That is, while components can be solved for by doing an *eigenvalue* or *singular value decomposition*, factors are estimated as best fitting solutions (Eckart and Young, 1936; Householder and Young, 1938), normally through iterative methods (Jöreskog, 1978; Lawley and Maxwell, 1963). Cattell (1965) referred to components analysis as a closed model and factor analysis as an open model, in that by explaining just the common variance, there was still more variance to explain.

Why is factor analysis controversial? At the structural level (i.e., the covariance or correlation matrix), there are normally more observed variables than parameters to estimate and the procedure is merely one of finding the best fitting solution using *ordinary least squares*, *weighted least squares*, or *maximum likelihood* (Jöreskog, 1978; Lawley and Maxwell, 1963). But at the data level, as Schonemann and Steiger (1978); Schonemann (1990); Steiger (1990) have shown, the model is indeterminate, although scores can be estimated. Velicer and Jackson (1990b,a) provide an extensive discussion of the differences between the two models and argue for the utility of PCA. Commentaries defending FA emphasize the benefits of FA for theory construction and evaluation (Loehlin, 1990; McArdle, 1990; Preacher and MacCallum, 2003).

The factor model partitions the correlation or covariance matrix into that which is predictable by *common factors*, $\mathbf{FF'}$, and that which is unique, $\mathbf{U}^2$ (the diagonal matrix of *uniquenesses*). Within R there are at several main algorithms that are used to estimate the factor coefficients. *maximum likelihood* may be done using the `factanal` function, or the

Table 6.3 First principal component of the **R** matrix. The solution, although capturing the rank order of how important the variables are, underestimates the highest ones, over estimates the lowest ones, and fails to account for diagonal of the matrix. This is shown most clearly by considering the matrix of residuals (**R** - **CC**′, the data - the model). Compare these residuals to those found using principal axes factor analysis as seen in Table 6.4. The output of `principal` is an object of type `psych` to allow for comparisons with factor analysis output from the factor analysis function, `fa`. The the uniquenesses are $diag(\mathbf{R} - \mathbf{CC}')$ and the loadings are correlations of the components with the variables. It is important to note that the recovered loadings do not match the factor loadings that were used to generate the data (Table 6.1).

```
> pc1 <- principal(R,1)
> pc1

Uniquenesses:
   V1    V2    V3    V4    V5    V6
0.220 0.307 0.408 0.519 0.635 0.748
Loadings:
   PC1
V1 0.88
V2 0.83
V3 0.77
V4 0.69
V5 0.60
V6 0.50
                 PC1
SS loadings    3.142
Proportion Var 0.524

> round(pc1$loadings %*% t(pc1$loadings),2) #show  the model

    V1   V2   V3   V4   V5   V6
V1 0.77 0.73 0.68 0.61 0.53 0.44
V2 0.73 0.69 0.64 0.57 0.50 0.42
V3 0.68 0.64 0.59 0.53 0.46 0.38
V4 0.61 0.57 0.53 0.48 0.41 0.34
V5 0.53 0.50 0.46 0.41 0.36 0.30
V6 0.44 0.42 0.38 0.34 0.30 0.25

> Rresid <-  R - pc1$loadings %*% t(pc1$loadings) #find the residuals
> round(Rresid,2)

      V1    V2    V3    V4    V5    V6
V1  0.23 -0.01 -0.05 -0.07 -0.08 -0.08
V2 -0.01  0.31 -0.08 -0.09 -0.10 -0.09
V3 -0.05 -0.08  0.41 -0.11 -0.11 -0.10
V4 -0.07 -0.09 -0.11  0.52 -0.11 -0.10
V5 -0.08 -0.10 -0.11 -0.11  0.64 -0.10
V6 -0.08 -0.09 -0.10 -0.10 -0.10  0.75
```

`fa` function in the **psych** package. `fa` will also provide a *principal axes*, *minimum residual*, *weighted least squares* or *generalized least squares* solution. Several other algorithms are available in packages such as *FAiR* and even more are possible but have not yet been implemented into R.

6.2.1 Principal Axes Factor Analysis as an eigenvalue decomposition of a reduced matrix

Principal components represents a $n*n$ matrix in terms of the first k components. It attempts to reproduce all of the **R** matrix. *Factor analysis* on the other hand, attempts to model just the common part of the matrix, which means all of the off-diagonal elements and the common part of the diagonal (the *communalities*). The non-common part, the *uniquenesses*, are simply that which is left over. An easy to understand procedure is *principal axes* factor analysis. This is similar to principal components, except that it is done with a reduced matrix where the diagonals are the communalities. The communalities can either be specified a priori, estimated by such procedures as multiple linear regression, or found by iteratively doing an eigenvalue decomposition and repeatedly replacing the original 1s on the diagonal with the the value of 1 - u^2 where

$$\mathbf{U}^2 = diag(\mathbf{R} - \mathbf{F}\mathbf{F}').$$

That is, starting with the original correlation or covariance matrix, **R**, find the k largest principal components, reproduce the matrix using those principal components. Find the resulting residual matrix, $\mathbf{R}^*$ and uniqueness matrix, $\mathbf{U}^2$ by

$$\mathbf{R}^* = \mathbf{R} - \mathbf{F}\mathbf{F}' \tag{6.5}$$

$$\mathbf{U}^2 = diag(\mathbf{R}^*)$$

and then, for iteration i, find $\mathbf{R}_i$ by replacing the diagonal of the original **R** matrix with 1 - diag($\mathbf{U}^2$) found on the previous step. Repeat this process until the change from one iteration to the next is arbitrarily small. This procedure is implemented in the `fa` and `factor.pa` functions in the **psych** package. (See Table 6.4 for an example of how well a single factor can reproduce the original correlation matrix. The eggs have been unscrambled.) It is a useful exercise to run `fa` using the principal axis factor method (fm= "pa") and specifying the number of iterations (e.g., max.iter=2). Then examine the size of the residuals as the number of iterations increases. When this is done, the solution gets progressively better, in that the size of the residuals in the off diagonal matrix become progressively smaller.

Rather than starting with initial communality estimates of 1, the process can be started with other estimates of the communality. A conventional starting point is the lower bound estimate of the communalities, the *squared multiple correlation* or *SMC* (Roff, 1936). The concept here is that a variable's communality must be at least as great as the amount of its variance that can be predicted by all of the other variables. A simple proof that the SMC is a lower bound for the communality was given by Harris (1978). A somewhat greater lower bound has been proposed by Yanai and Ichikawa (1990). The squared multiple correlations of each variable with the remaining variables are the diagonal elements of

$$\mathbf{I} - (diag(\mathbf{R}^{-1})^{-1}$$

and thus a starting estimate for $\mathbf{R_0}$ would be $\mathbf{R} - (diag(\mathbf{R}^{-1})^{-1}$.

It is interesting to recognize that the obverse of this relationship holds and the communality of a variable is the upper bound of the squared multiple correlation with that variable. To increase the predictability of a variable, **x**, from a set of other variables, **Y**, it does not help to add variables to the **Y** set that load on factors already measured, for this will not change the communality. It is better to find variables that correlate with **x** but measure factors not already included in **Y** (Guttman, 1940). Tryon (1957) provides a detailed discussion of multiple ways to estimate a variable's communality, with particular reference to domain sampling models of reliability. For most problems, the initial communality estimate does not have an effect upon the final solution, but in some cases, adjusting the start values to be 1 rather than the SMC will lead to better solutions when doing principal axes decompositions.

At least three indices of goodness of fit of the principal factors model can be considered: One compares the sum of squared residuals to the sum of the squares of the original values:

$$GF_{total} = 1 - \frac{\mathbf{1R^{*2}1'}}{\mathbf{1R^{2}1'}}$$

The second does the same, but does not consider the diagonal of **R**

$$GF_{offdiagonal} = 1 - \frac{\sum_{i \neq j} r_{ij}^{*2}}{\sum_{i \neq j} r_{ij}^{*2}} = 1 - \frac{\mathbf{1R^{*2}1'} - tr(\mathbf{1R^{*2}1'}}{\mathbf{1R^{2}1'} - tr(\mathbf{1R^{2}1'})}$$

That is, if the sum is taken over all $i \neq j$, then this evaluates how well the factor model fits the off diagonals, if the sum includes the diagonal elements as well, then the GF statistic is evaluating the overall fit of the model to the data. Finally, a χ^2 test of the size of the residuals simply sums all the squared residuals and multiplies by the number of observations:

$$\chi^2 = \sum_{i<j} r*_{ij}^2 (N-1)$$

with p * (p-1)/2 degrees of freedom.

6.2.2 Maximum Likelihood Factor Analysis and its alternatives

The fundamental factor equation (Equation 6.3) may be viewed as set of simultaneous equations which may be solved several different ways: *ordinary least squares*, *generalized least squares*, and *maximum likelihood*. Ordinary least squares (*OLS*) or *unweighted least squares* (*ULS*) minimizes the sum of the squared residuals when modeling the sample correlation or covariance matrix, **S**, with $\Sigma = \mathbf{FF'} + \mathbf{U^2}$

$$E = \frac{1}{2} tr(\mathbf{S} - \Sigma)^2 \tag{6.6}$$

where the *trace*, `tr`, of a matrix is the sum of the diagonal elements and the division by two reflects the symmetry of the **S** matrix. As discussed by Jöreskog (1978) and Loehlin (2004), Equation 6.6 can be generalized to weight the residuals $(\mathbf{S} - \Sigma)$ by the inverse of the sample matrix, **S**, and thus to minimize

Table 6.4 Principal Axis Factor Analysis with one factor. The solution was iterated until the communality estimates (1- the uniquenesses) failed to change from one iteration to the next. Notice how the off diagonal of the residual matrix is effectively zero. Compare this with the solution of principal components seen in Table 6.3.

```
> f1 <- factor.pa(R)
> f1

Factor Analysis using method =  pa
Call: factor.pa(r = R)
  V PA1   h2   u2
1 1 0.9 0.81 0.19
2 2 0.8 0.64 0.36
3 3 0.7 0.49 0.51
4 4 0.6 0.36 0.64
5 5 0.5 0.25 0.75
6 6 0.4 0.16 0.84

                PA1
SS loadings    2.71
Proportion Var 0.45

Test of the hypothesis that 1 factor is sufficient.

The degrees of freedom for the model is 9  and the fit was  0

Measures of factor score adequacy             PA1
 Correlation of scores with factors           0.94
Multiple R square of scores with factors      0.89
Minimum correlation of factor score estimates 0.78
Validity of unit weighted factor scores       0.91


> f1$loadings %*% t(f1$loadings) #show the model

     V1   V2   V3   V4   V5   V6
V1 0.81 0.72 0.63 0.54 0.45 0.36
V2 0.72 0.64 0.56 0.48 0.40 0.32
V3 0.63 0.56 0.49 0.42 0.35 0.28
V4 0.54 0.48 0.42 0.36 0.30 0.24
V5 0.45 0.40 0.35 0.30 0.25 0.20
V6 0.36 0.32 0.28 0.24 0.20 0.16

> Rresid <- R - f1$loadings %*% t(f1$loadings)  #show the residuals
> round(Rresid,2)

     V1   V2   V3   V4   V5   V6
V1 0.19 0.00 0.00 0.00 0.00 0.00
V2 0.00 0.36 0.00 0.00 0.00 0.00
V3 0.00 0.00 0.51 0.00 0.00 0.00
V4 0.00 0.00 0.00 0.64 0.00 0.00
V5 0.00 0.00 0.00 0.00 0.75 0.00
V6 0.00 0.00 0.00 0.00 0.00 0.84
```

$$E = \frac{1}{2} tr((\mathbf{S} - \Sigma)\mathbf{S}^{-1})^2 = \frac{1}{2} tr(\mathbf{I} - \Sigma\mathbf{S}^{-\mathbf{1}})^2. \tag{6.7}$$

This is known as *generalized least squares* (*GLS*) or *weighted least squares* (*WLS*). Similarly, if the residuals are weighted by the inverse of the model, Σ, minimizing

$$E = \frac{1}{2} tr((\mathbf{S} - \Sigma)\Sigma^{-1})^2 = \frac{1}{2} tr(\mathbf{S}\Sigma^{-1} - I)^2 \tag{6.8}$$

will result in a model that maximizes the likelihood of the data. This procedure, *maximum likelihood estimation* (*MLE*) is also seen as finding the minimum of

$$E = \frac{1}{2} \left(tr(\Sigma^{-1}\mathbf{S}) - ln \left| \Sigma^{-1}\mathbf{S} \right| - p \right) \tag{6.9}$$

where p is the number of variables (Jöreskog, 1978). Perhaps a helpful intuitive explanation of Equation 6.9 is that if the model is correct, then $\Sigma = \mathbf{S}$ and thus $\Sigma^{-1}\mathbf{S} = \mathbf{I}$. The trace of an identity matrix of rank p is p, and the logarithm of $|\mathbf{I}|$ is 0. Thus, the value of E if the model has perfect fit is 0. With the assumption of multivariate normality of the residuals, and for large samples, a χ^2 statistic can be estimated for a model with p variables and f factors (Bartlett, 1951; Jöreskog, 1978; Lawley and Maxwell, 1962):

$$\chi^2 = \left(tr(\Sigma^{-1}\mathbf{S}) - ln \left| \Sigma^{-1}\mathbf{S} \right| - p \right) (N - 1 - (2p + 5)/6 - (2f)/3) . \tag{6.10}$$

This χ^2 has degrees of freedom:

$$df = p * (p - 1)/2 - p * f + f * (f - 1)/2. \tag{6.11}$$

That is, the number of lower off-diagonal correlations - the number of unconstrained loadings (Lawley and Maxwell, 1962).

6.2.2.1 Minimal Residual Factor Analysis

All of the previous factor analysis procedures attempt to optimize the fit of the model matrix (Σ) to the correlation or covariance matrix ($\mathbf{S}$). The diagonal of the matrix is treated as mixture of common variance and unique variance and the problem becomes one of estimating the common variance (the *communality* of each variable). An alternative is to ignore the diagonal and to find that model which minimizes the squared residuals of the off diagonal elements (Eckart and Young, 1936; Harman and Jones, 1966). This is done in the `fa` function using the "minres" (default) option by finding the solution that minimizes

$$\frac{1}{2} \mathbf{1} ((\mathbf{S} - \mathbf{I}) - (\Sigma - tr(\Sigma))^2 \mathbf{1}'. \tag{6.12}$$

The advantage of the *minres* solution is that it does not require finding the inverse of either the original correlation matrix (as do *GLS* and *WLS*) nor of the model matrix (as does *MLE*, and thus can be performed on non-positive definite matrices or matrices that are not invertible.

6.2.2.2 Factor analysis by using the `optim` function

Factor analysis is one example of an iterative search to minimize a particular criterion. The easiest way of doing this is to use the `optim` function which will find a vector of values that minimize or maximize a particular criterion. If the function has an analytic derivative, the optimization will be much faster. The `fa` function in *psych* uses `optim` to do factor analysis optimizing most of the various criteria discussed above (all except principal axis which is done by direct iteration). Conceptually, all of these criteria find the values of the communalities that produce an eigen value decomposition that minimizes a particular definition of error. That is, the fitting procedure for *minimum residual factor analysis* minimizes the squared off diagonal elements of the residual matrix, *ordinary least squares* minimizes the error in Equation 6.6, *weighted least squares* minimizes the error in Equation 6.7, and *maximum likelihood* minimizes the error in Equations 6.8 or 6.9. The minimization function in `fa` is taken (almost directly) from the fitting function used in `factanal` and generalized for the *OLS* and *GLS* methods.

6.2.3 Comparing extraction techniques

Assuming that the residual variance reflects normally distributed random error, the most elegant statistical solution is that of maximum likelihood (Lawley and Maxwell, 1963). If the residuals are not normally distributed, for instance if they represent many individual nuisance factors (MacCallum and Tucker, 1991), then maximum likelihood techniques do not recreate the major factor pattern nearly as well as ordinary least squares techniques (MacCallum et al., 2007). Other times that the MLE technique is not recommended is when it will not converge or when the data are particularly noisy. Many demonstrations of factor structures assume that except for the major factors, all residuals are normally distributed around 0. This is the structure assumed when doing ML factor analysis. An alternative, and perhaps more realistic situation, is that the there are a few major (big) factors and many minor (small) factors. The challenge is thus to identify the major factors. `sim.minor` generates such structures. The structures generated can be thought of as having a a major factor structure with some small correlated residuals.

However, in the normal case, factor analysis using either maximum likelihood in `factanal` or `fa` functions is probably most appropriate. In particular, `factanal` can use multiple start values and iterates to solutions that avoid Heywood (1931) cases of negative uniquenesses. An example of a one factor solution is shown in Table 6.5. For the data in Table 6.1 it is necessary to specify that the input is from covariance or correlation matrix rather than from a raw data matrix. The number of factors to extract is specified at 1. Table 6.5 shows the factor loadings, uniquenesses, and goodness of fit. The number of observations was arbitrarily set at 100 for this demonstration. The loadings are the correlations of the tests or items with the factors. The sum of the squared loadings for the factor is the eigenvalue of the factor and is the sum across all the items of the amount of variance accounted for by the factor. The uniquenesses are $u_i = 1 - h_i^2$ where the communalities, h^2 are the elements of the diagonal of $\mathbf{FF'}$ The χ^2 measure of fit is perfect (because the correlation matrix was in fact generated from the factor matrix with no error). The probability is 1 because the fit is perfect.

The *minimum residual* and *principal axes* models do not require finding the inverse of either the original correlation matrix, $\mathbf{R}$, or of the model matrix, $\mathbf{FF'}$. As a consequence they

Table 6.5 One factor maximum likelihood solution to the correlation matrix created in Table 6.1 using maximum likelihood estimation by the `fa` function. The number of observations is set arbitrarily to 100 to demonstrate how to run the function.

```
> fm1 <- fa(R,nfactors=1,n.obs=100,fm="ml")
> fm1

Factor Analysis using method =  ml
Call: fa(r = R, nfactors = 1, n.obs = 100, fm = "ml")
  V MR1   h2   u2
1 1 0.9 0.81 0.19
2 2 0.8 0.64 0.36
3 3 0.7 0.49 0.51
4 4 0.6 0.36 0.64
5 5 0.5 0.25 0.75
6 6 0.4 0.16 0.84

                MR1
SS loadings    2.71
Proportion Var 0.45

Test of the hypothesis that 1 factor is sufficient.

The degrees of freedom for the model is 9  and the fit was  0
The number of observations was  100  with Chi Square =  0  with prob <  1

Measures of factor score adequacy             MR1
 Correlation of scores with factors           0.94
Multiple R square of scores with factors      0.89
Minimum correlation of factor score estimates 0.78
Validity of unit weighted factor scores       0.91
```

will produce solutions for singular matrices where the *maximum likelihood* or *generalized least squares* or *weighted least squares* will not. The `Harman.Burt` correlation matrix of eight emotional variables in the `Harman` data set is an example of where a *minres* solution will work but a *maximum likelihood* one will not.

The *principal axes* algorithm in the *psych* iteratively estimates the communalities by repeatedly doing an *eigen value decomposition* of the original correlation matrix where the diagonal values are replaced by the estimated values from the previous iteration. When these values fail to change, the algorithm terminates. This differs from the principal axes method discussed by Harman (1976) which starts with an initial estimate of communalities and does not iterate. The iterative solution matches what Harman (1976) refers to as a *minres* solution and what SPSS calls a *ULS* solution. The *minres* solution in the `fa` function seems to more closely approximate that of a *mle* solution.

6.2.4 Exploratory analysis with more than one factor/component

Originally developed by Spearman (1904a) to examine one factor of ability, factor analysis is now routinely used for the case of more than one underlying factor. Consider the correlation

matrix in Table 6.6 generated by a two factor model. The correlation matrix was created by matrix multiplication of a factor matrix, $\mathbf{F}$ with its transpose, $\mathbf{R} = \mathbf{F}\mathbf{F}'$ and then labels were added using the `rownames` function. A much simpler way would have been to use the `sim` function. Although not shown, fitting one factor/component models to this correlation matrix provides a very poor solution, and fitting a two component model (Table 6.6) once again tends to overestimate the effects of the weaker variables. The residuals are found by calculation ($\mathbf{R}^* = \mathbf{R} - \mathbf{C}\mathbf{C}'$ but could have been found by requesting the residuals from the output of `principal`, Rresid = pc2$residual. Fitting a two factor model estimated by `fa` correctly recreates the pattern matrix (Table 6.7).

6.2.5 Comparing factors and components- part 1

Although on the surface, the component model and factor model appear to very similar (compare Tables 6.6 and 6.7), they are logically very different. In the components model, components are linear sums of the variables. In the factor model, on the other hand, factors are latent variables whose weighted sum accounts for the common part of the observed variables. In path analytic terms, for the component model, arrows go from the variables to the components, while in the factor model they go from the factors to the variables (Figure 6.1). Logically this implies that the addition of one or more variables into a factor space should not change the factors, but should change the components.

An example of this is when two additional variables are added to the correlation matrix (Table 6.8). The factor pattern does not change for the previous variables, but the component pattern does. Why is this? Because the components are aimed at accounting for all of the variance of the matrix, adding new variables increases the amount of variance to be explained and changes the previous estimates. But the common part of the variables (that which is estimated by factors) is not sensitive to the presence (or absence) of other variables. This addition of new variables should make a larger difference when there are only a few variables representing a factor.

The difference between components and factors may also be seen by examining the effect of the number of markers per factor/component when loadings are plotted as a function of the number of variables defining a factor/component. For five uni-dimensional correlation matrices with correlations ranging from .05 to .55 (and thus factor loadings ranging from $\sqrt{.05}$ to $\sqrt{.55}$) and the number of variables ranging from 2 to 20, the factor loadings in all cases are exactly the square root of the between item correlations. But the component loadings are a function of the correlations (as they should be) and of the number of variables. The component loadings asymptotically tend towards the corresponding factor loadings (Figure 6.2). With fewer than five to ten variables per component, and particularly for low communality items, the inflation in the component loadings is substantial. Although a fundamental difference between the two models, this problem of the additional variable is most obvious when there are not very many markers for each factor and becomes less of an empirical problem as the number of variables per component increases (Widaman, 1993).

Table 6.6 A somewhat more complicated model than Table tab:fa:pa has two factors. The analysis examines the first two principal components. Compare this with the analysis of the the first two factors using *minimum residual* factor analysis (Table 6.7). Both solutions are tested by trying to recreate the original matrix. Examine the size of the residual matrix, **Rresid**. A simpler way to construct the data matrix would have been to use the `sim` function.

```
>F  <-  matrix(c(.9,.8,.7,rep(0,6),.8,.7,.6),ncol=2) #the model
> rownames(F) <- paste("V",seq(1:6),sep="")  #add labels
> colnames(F <- c("F1", "F2")
> R <- F %*% t(F) #create the correlation matrix
> diag(R) <- 1  #adjust the diagonal of the matrix
> R

     V1   V2   V3   V4   V5   V6
V1 1.00 0.72 0.63 0.00 0.00 0.00
V2 0.72 1.00 0.56 0.00 0.00 0.00
V3 0.63 0.56 1.00 0.00 0.00 0.00
V4 0.00 0.00 0.00 1.00 0.56 0.48
V5 0.00 0.00 0.00 0.56 1.00 0.42
V6 0.00 0.00 0.00 0.48 0.42 1.00

> pc2 <- principal(R,2)
> pc2

Uniquenesses:
   V1    V2    V3    V4    V5    V6
0.182 0.234 0.309 0.282 0.332 0.409
Loadings:
   PC1  PC2
V1 0.90
V2 0.88
V3 0.83
V4      0.85
V5      0.82
V6      0.77
                 PC1   PC2
SS loadings    2.273 1.988
Proportion Var 0.379 0.331
Cumulative Var 0.379 0.710

> round(pc2$loadings %*% t(pc2$loadings),2)

   V1   V2   V3   V4   V5   V6
V1 0.81 0.79 0.75 0.00 0.00 0.00
V2 0.79 0.77 0.73 0.00 0.00 0.00
V3 0.75 0.73 0.69 0.00 0.00 0.00
V4 0.00 0.00 0.00 0.72 0.70 0.65
V5 0.00 0.00 0.00 0.70 0.67 0.63
V6 0.00 0.00 0.00 0.65 0.63 0.59

> Rresid <- R - pc2$loadings %*% t(pc2$loadings)
> round(Rresid,2)

      V1    V2    V3    V4    V5    V6
V1  0.19 -0.07 -0.12  0.00  0.00  0.00
V2 -0.07  0.23 -0.17  0.00  0.00  0.00
V3 -0.12 -0.17  0.31  0.00  0.00  0.00
V4  0.00  0.00  0.00  0.28 -0.14 -0.17
V5  0.00  0.00  0.00 -0.14  0.33 -0.21
V6  0.00  0.00  0.00 -0.17 -0.21  0.41
```

Table 6.7 Comparing a two factor with the two component solution of Table 6.6. This analysis examines the first two factors. Compare this with the analysis of the the first two principal components (Tabel 6.6). Both solutions are tested by trying to recreate the original matrix. As should be the case, the diagonal residuals are larger, but the off diagonal residuals are smaller for the factor solution.

```
> f2 <- fa(R,2)
> f2
> round(f2$loadings %*% t(f2$loadings),2)

Factor Analysis using method =  minres
Call: fa(r = R, nfactors = 2)
   V MR1 MR2   h2   u2
V1 1 0.9     0.81 0.19
V2 2 0.8     0.64 0.36
V3 3 0.7     0.49 0.51
V4 4     0.8 0.64 0.36
V5 5     0.7 0.49 0.51
V6 6     0.6 0.36 0.64

                MR1  MR2
SS loadings    1.94 1.49
Proportion Var 0.32 0.25
Cumulative Var 0.32 0.57

Test of the hypothesis that 2 factors are sufficient.

The degrees of freedom for the model is 4  and the fit was  0

Measures of factor score adequacy             MR1 MR2
 Correlation of scores with factors           0.94 0.88
Multiple R square of scores with factors      0.88 0.77
Minimum correlation of factor score estimates 0.75 0.53
Validity of unit weighted factor scores       0.92 0.86

     V1   V2   V3   V4   V5   V6
V1 0.81 0.72 0.63 0.00 0.00 0.00
V2 0.72 0.64 0.56 0.00 0.00 0.00
V3 0.63 0.56 0.49 0.00 0.00 0.00
V4 0.00 0.00 0.00 0.64 0.56 0.48
V5 0.00 0.00 0.00 0.56 0.49 0.42
V6 0.00 0.00 0.00 0.48 0.42 0.36

> round(f2$residual,2)

     V1   V2   V3   V4   V5   V6
V1 0.19 0.00 0.00 0.00 0.00 0.00
V2 0.00 0.36 0.00 0.00 0.00 0.00
V3 0.00 0.00 0.51 0.00 0.00 0.00
V4 0.00 0.00 0.00 0.36 0.00 0.00
V5 0.00 0.00 0.00 0.00 0.51 0.00
V6 0.00 0.00 0.00 0.00 0.00 0.64
```

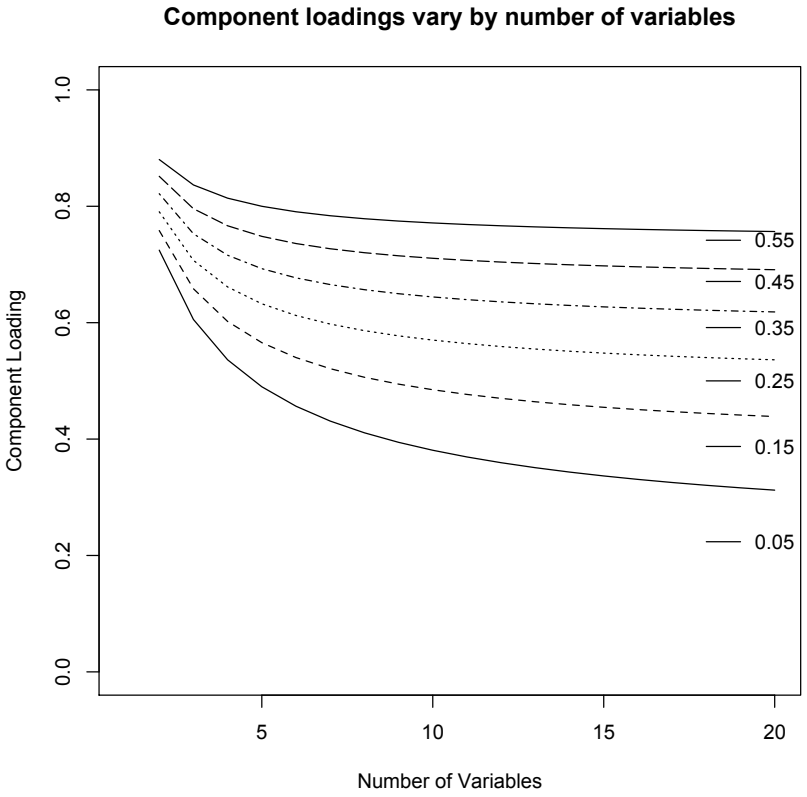

Fig. 6.2 Component loadings are a function both of the item correlations and the number of variables defining the component. The curved lines represent the component loadings for matrices with correlations ranging from .05 (bottom line) to .55 (top line). The asymptotic value of the loadings is the factor loadings or the square root of the corresponding correlation (shown on the right hand side). With fewer than five to ten variables per component, the inflation in the component loadings is substantial.

6.3 Rotations and Transformations

The original solution of a principal components or principal axes factor analysis is a set of vectors that best account for the observed covariance or correlation matrix, and where the components or factors account for progressively less and less variance. But such a solution, although maximally efficient in describing the data, is rarely easy to interpret. But what makes a structure easy to interpret? Thurstone's answer, *simple structure*, consists of five rules (Thurstone, 1947, p 335):

(1) Each row of the oblique factor matrix **V** should have at least one zero.

(2) For each column p of the factor matrix **V** there should be a distinct set of r linearly independent tests whose factor loadings v_{ip} are zero.

(3) For every pair of columns of **V** there should be several tests whose entries v_{ip} vanish in one column but not in the other.

Table 6.8 Comparing factors and components when an additional variable is added. Comparing these results to Table 6.7, it is important to note that the loadings of the old items remain the same.

```
> f  <-  matrix(c(.9,.8,.7,rep(0,3),.7,rep(0,4),.8,.7,.6,0,.5),ncol=2) #the model
> rownames(f) <- paste("V",seq(1:8),sep="")  #add labels
>  colnames(f) <- c("F1", "F2")
>  R <- f %*% t(f) #create the correlation matrix
>  diag(R) <- 1  #adjust the diagonal of the matrix
>  R

     V1   V2   V3   V4   V5   V6   V7   V8
V1 1.00 0.72 0.63 0.00 0.00 0.00 0.63 0.00
V2 0.72 1.00 0.56 0.00 0.00 0.00 0.56 0.00
V3 0.63 0.56 1.00 0.00 0.00 0.00 0.49 0.00
V4 0.00 0.00 0.00 1.00 0.56 0.48 0.00 0.40
V5 0.00 0.00 0.00 0.56 1.00 0.42 0.00 0.35
V6 0.00 0.00 0.00 0.48 0.42 1.00 0.00 0.30
V7 0.63 0.56 0.49 0.00 0.00 0.00 1.00 0.00
V8 0.00 0.00 0.00 0.40 0.35 0.30 0.00 1.00

> f2 <- factanal(covmat=R,factors=2)
> f2

Call:
factanal(factors = 2, covmat = R)
Uniquenesses:
  V1   V2   V3   V4   V5   V6   V7   V8
0.19 0.36 0.51 0.36 0.51 0.64 0.51 0.75
Loadings:
   Factor1 Factor2
V1 0.9
V2 0.8
V3 0.7
V4         0.8
V5         0.7
V6         0.6
V7 0.7
V8         0.5
               Factor1 Factor2
SS loadings      2.430   1.740
Proportion Var   0.304   0.218
Cumulative Var   0.304   0.521
The degrees of freedom for the model is 13 and the fit was 0
```

(4) For every pair of columns of **V**, a large proportion of the tests should have zero entries in both columns. This applies to factor problems with four or five or more common factors.

(5) For every pair of columns there should preferably be only a small number of tests with non-vanishing entries in both columns.

Thurstone proposed to rotate the original solution to achieve simple structure.

A matrix is said to be *rotated* if it is multiplied by a matrix of orthogonal vectors that preserves the communalities of each variable. Just as the original matrix was orthogonal, so is the rotated solution. For two factors, the *rotation* matrix **T** will rotate the two factors $\boldsymbol{\theta}$ radians in a counterclockwise direction.

Table 6.9 Two principal components with two additional variables. Compared to Table 6.6, the loadings of the old variables have changed when two new variables are added.

```
pc2 <- principal(R,2)
pc2

Uniquenesses:
   V1    V2    V3    V4    V5    V6    V7    V8
0.194 0.271 0.367 0.311 0.379 0.468 0.367 0.575
Loadings:
   PC1  PC2
V1 0.90
V2 0.85
V3 0.80
V4      0.83
V5      0.79
V6      0.73
V7 0.80
V8      0.65
                 PC1   PC2
SS loadings    2.812 2.268
Proportion Var 0.352 0.284
Cumulative Var 0.352 0.635
```

$$T = \begin{pmatrix} cos(\theta) & sin(\theta) \\ -sin(\theta) & cos(\theta) \end{pmatrix} \tag{6.13}$$

Generalizing equation 6.13 to larger matrices is straight forward:

$$T = \begin{pmatrix} 1 & \dots & 0 & \dots & 0 & \dots & 0 \\ 0 & \dots & cos(\theta) & \dots & sin(\theta) & \dots & 0 \\ \dots & \dots & 0 & 1 & 0 & \dots & 0 \\ 0 & \dots & -sin(\theta) & \dots & cos(\theta) & \dots & 0 \\ \dots & \dots & 0 & \dots & 0 & \dots & \dots \\ 0 & \dots & 0 & \dots & 0 & \dots & 1 \end{pmatrix}. \tag{6.14}$$

When $\mathbf{F}$ is post-multiplied by $\mathbf{T}$, $\mathbf{T}$ will rotate the i^{th} and j^{th} columns of $\mathbf{F}$ by θ radians in a counterclockwise direction.

$$\mathbf{F}_r = \mathbf{FT} \tag{6.15}$$

The `factor.rotate` function from the **psych** package will do this rotation for arbitrary angles (in degrees) for any pairs of factors. This is useful if there is a particular rotation that is desired.

As pointed out by Carroll (1953) when discussing Thurstone's (1947) simple structure as a rotational criterion "it is obvious that there could hardly be any single mathematical expression which could embody all these characteristics." (p 24). Carroll's solution to this was to minimize the sum of the inner products of the squared (rotated) loading matrix. An alternative, discussed by Ferguson (1954) is to consider the *parsimony* of a group of n tests with r factors to be defined as the average parsimony of the individual tests (I_j) where

$$I_j = \sum_m^r a_{jm}^4 \tag{6.16}$$

(the squared communality) and thus the average parsimony is

$$I_. = n^{-1} \sum_j^n \sum_m^r a_{jm}^4$$

and to choose a rotation that maximizes parsimony.

Parsimony as defined in equation 6.16 is a function of the variance as well as the mean of the squared loadings of a particular test on all the factors. For fixed communality h^2, it will be maximized if all but one loading is zero; a variable's parsimony will be maximal if one loading is 1.0 and the rest are zero. In path notation, parsimony is maximized if one and only one arrow is associated with a variable. This criterion, as well as the criterion of maximum variance taken over factors has been operationalized as the *quartimax*[4] criterion by Neuhaus and Wrigley (1954). As pointed out by Kaiser (1958), the criterion can rotate towards a solution with one general factor, ignoring other, smaller factors.

If a general factor is not desired, an alternative measure of the parsimony of a factor, similar to equation 6.16 is to maximize the variance of the squared loadings taken over items instead of over factors. This, the *varimax* criterion was developed by Kaiser (1958) to avoid the tendency to yield a general factor. Both of these standard rotations as well as many others are available in the **GPArotation** package of rotations and transformations which uses the *Gradient Projection Algorithms* developed by Jennrich (2001, 2002, 2004).

6.3.1 Orthogonal rotations

Consider the correlation matrix of eight physical measurements reported by Harman (1976) and included as the `Harman23.cor` data set in base R. Factoring this set using `factanal` and not rotating the solution produces the factor loadings shown in the left panel of Figure 6.3 and in Table 6.10. Rotating using the *varimax* criterion (Kaiser, 1958) produces the solution shown in the right hand panel of Figure 6.3 and in Table 6.11 Kaiser (1958). An alternative way to represent these solutions is in terms of path diagrams (Figure 6.4). Here, all loadings less than .30 are suppressed.

6.3.2 Oblique transformations

Many of those who use factor analysis use it to identify theoretically meaningful constructs which they have no reason to believe are orthogonal. This has lead to the use of *oblique transformations* which allow the factors to be correlated. (The term oblique is used for historical reasons regarding the use of reference factors which became oblique as the angles between the factors themselves become acute.) Although the term rotation is sometimes used

[4] The original programs for doing this and other rotations and transformations were written in FORTRAN and tended to use all caps. Thus, one will see QUARTIMAX, VARIMAX, PROMAX as names of various criteria. More recent usage has tended to drop the capitalization.

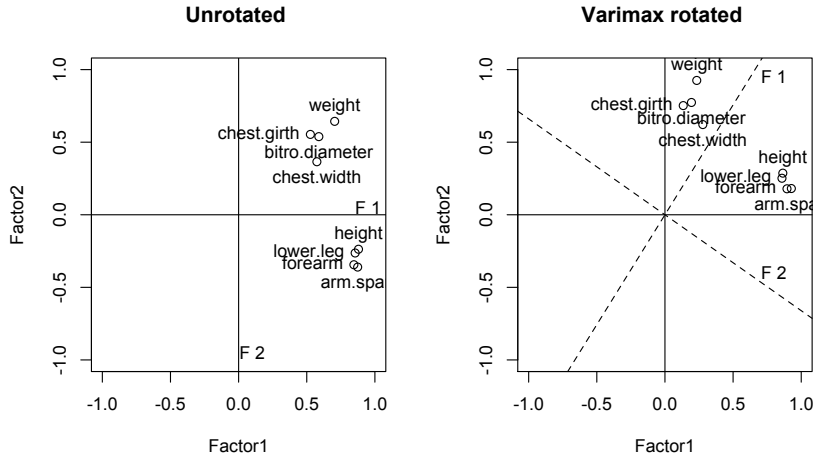

Fig. 6.3 Comparing unrotated and varimax rotated solutions for 8 physical variables from Harman (1976). The first factor in the unrotated solution has the most variance but the structure is not "simple", in that all variables have high loadings on both the first and second factor. The first factor seems to represent overall size, the second factor is a bipolar factor separating height from width. The varimax rotation provides a simpler solution with factors easily interpreted as "lankiness" (factor 1) and "stockiness" (factor 2). The rotation angle was 33.45 degrees clockwise. The original axes from the left panel are shown as dashed lines in the right panel. Compare this figure with the path representation in Figure 6.4.

for both *orthogonal* and *oblique* solutions, in the oblique case the factor matrix is not rotated so much as *transformed*.

Oblique transformations lead to the distinction between the *factor pattern* and *factor structure* matrices. The *factor pattern* matrix is the set of *regression weights* (loadings) from the latent factors to the observed variables (see equation 6.4). The *factor structure* matrix is the matrix of *correlations* between the factors and the observed variables. If the factors are uncorrelated, structure and pattern are identical. But, if the factors are correlated, the structure matrix ($\mathbf{S}$) is the pattern matrix ($\mathbf{F}$) times the factor intercorrelations ($\boldsymbol{\phi}$):

$$\mathbf{S} = \mathbf{F}\boldsymbol{\phi} <=> \mathbf{F} = \mathbf{S}\boldsymbol{\phi}^{-1} \tag{6.17}$$

and the modeled correlation matrix is the pattern matrix times the structure matrix plus the uniquenesses:

$$\mathbf{R} = \mathbf{F}\mathbf{S}' = \mathbf{F}\boldsymbol{\phi}'\mathbf{F}' + \mathbf{U}^2 = \mathbf{S}\boldsymbol{\phi}^{-1}\mathbf{S}' + \mathbf{U}^2 \tag{6.18}$$

Simple structure here really means "simple pattern", for the goal is to find a inter factor correlation matrix $\boldsymbol{\phi}$ and a transformation matrix $\mathbf{T}$ that have a simple pattern matrix matching the Thurstonian goals. An early solution to this problem was the *quartimin* algorithm developed by Carroll (1953) and then refined into the *biquartimin* Carroll (1957).

Alternatives to quartimin have included *promax* Hendrickson and White (1964) which transforms a varimax solution to a somewhat simpler but oblique target. The `Promax` function in *psych* is based upon the `promax` rotation option in `factanal` but returns the correlations between the factors. Promax finds the regression weights that will best transform the original

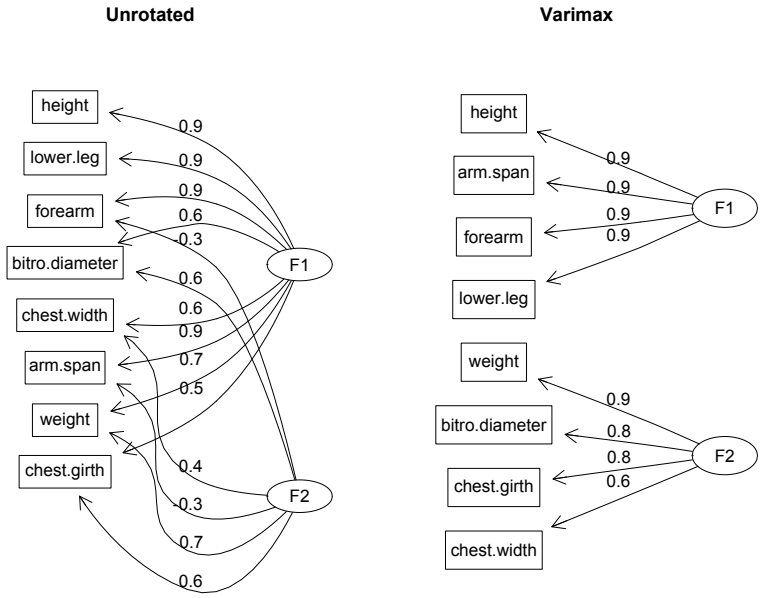

Fig. 6.4 An alternative way to show simple structure is in terms of a *path diagram*. Paths with coefficients less than .3 are not drawn. Although the two models fit equally well, the original solution (left panel) has more paths (is more complex) than the the *varimax* rotated solution to the right. Compare this path diagram to the spatial representation in Figure 6.3.

orthogonal loadings to the *varimax* rotated loadings raised to an arbitrary power (default of 4). More recent work in transformations has included the `simplimax` criterion (Kiers, 1994) which has been shown to do a good job of maximizing factor simplicity Lorenzo-Seva (2003). Available as part of the **GPArotation** package are rotations that maximize factorial simplicity using the criteria of Bentler (1977). Also available in the **GPArotation** package is the `geominQ` transformation which has proved useful in exploratory structural equation modeling (Asparouhov and Muthén, 2009) for it allows for solutions with some items having complexity two. Yet more alternative transformations towards specified targets are possible using the `target.rot` function. The default transformation for `target.rot` is towards an *independent cluster* solution in which each variable loads on one and only one factor.

Consider the factor solution to the Harman 8 physical variable problem shown earlier. By allowing the two dimensions to correlate, a simpler pattern matrix is achieved using an `oblimin` transformation., although at the cost of introducing a correlation matrix between the factors.

Table 6.10 Harman's 8 physical variables yield a clear unipolar size factor with a secondary, bipolar factor representing the distinction between height and width. The default rotation in `factanal` is varimax and is oblimin in `fa` and so it is necessary to specify "none" to get the unrotated solution.

```
> data(Harman23.cor)
> f2 <- factanal(factors=2,covmat=Harman23.cor,rotation="none")
> f2

Call:
factanal(factors = 2, covmat = Harman23.cor, rotation = "none")
Uniquenesses:
        height       arm.span        forearm      lower.leg         weight bitro.diameter    chest.girth
         0.170          0.107          0.166          0.199          0.089          0.364          0.416
   chest.width
         0.537
Loadings:
               Factor1 Factor2
height          0.880  -0.237
arm.span        0.874  -0.360
forearm         0.846  -0.344
lower.leg       0.855  -0.263
weight          0.705   0.644
bitro.diameter  0.589   0.538
chest.girth     0.526   0.554
chest.width     0.574   0.365

               Factor1 Factor2
SS loadings      4.434   1.518
Proportion Var   0.554   0.190
Cumulative Var   0.554   0.744

Test of the hypothesis that 2 factors are sufficient.
The chi square statistic is 75.74 on 13 degrees of freedom.
The p-value is 6.94e-11
```

6.3.3 Non-Simple Structure Solutions: The Simplex and Circumplex

In contrast to Thurstone's search for simple structure Thurstone (1947), Guttman (1954) discussed two structures that do not have simple structure, but that are seen in ability and interest tests. As discussed by Browne (1992), the *simplex* is a structure where it is possible to order tests such that the largest correlations are between adjacent tests, and the smallest correlations are between tests at the opposite end of the order. Patterns such as this are seen in measures taken over time, with the highest correlations between tests taken at adjacent times. Simplex patterns also occur when dichotomous items vary in difficulty. The resulting correlation matrix will tend to have the highest correlations between items with equal or near equal difficulty.

An alternative structure discussed by Guttman (1954) and Browne (1992) is the *circumplex*, where the pattern of correlations is such that the tests can be ordered so that the correlations between pairs first decreases and then increases. This is a common solution in the study of emotions and interpersonal relations. Introduced to the personality researcher by

Table 6.11 Varimax rotation applied to the 2 factor solution for 8 physical variables. Done using the `varimax` function in **GPArotation**.

```
> fv <- varimax(f2$loadings)   #loadings from prior analysis
> fv

Loadings:
               Factor1 Factor2
height         0.865   0.287
arm.span       0.927   0.181
forearm        0.895   0.179
lower.leg      0.859   0.252
weight         0.233   0.925
bitro.diameter 0.194   0.774
chest.girth    0.134   0.752
chest.width    0.278   0.621

               Factor1 Factor2
SS loadings      3.335   2.617
Proportion Var   0.417   0.327
Cumulative Var   0.417   0.744

$rotmat
           [,1]      [,2]
[1,]  0.8343961 0.5511653
[2,] -0.5511653 0.8343961
```

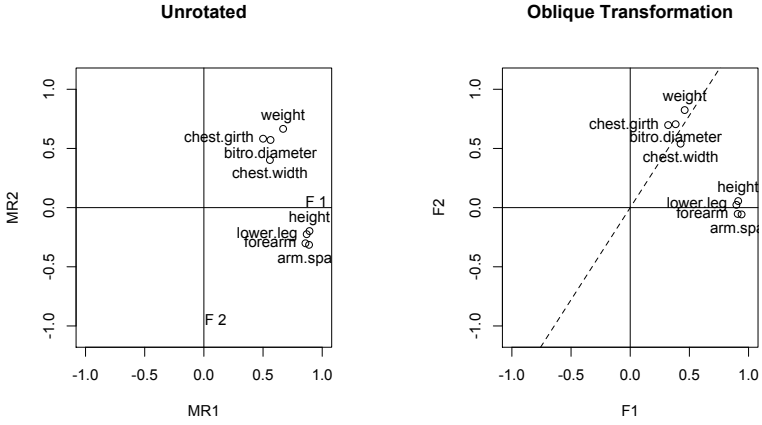

Fig. 6.5 An oblimin solution to the Harman physical variables problem. The left panel shows the original solution, the right panel shows the oblique solution. The dashed line shows the location of the transformed Factor 2.

Table 6.12 Factor or component solutions can be transformed using oblique transformations such as `oblimin`, `geominQ`, `Promax` or `target.rot`.The solution provides a simpler pattern matrix. The `print.psych` function defaults to show all loadings, but can be made to show a limited number of loadings by adjusting the "cut" parameter.

```
> f2t <- fa(Harman23.cor$cov,2,rotate="oblimin",n.obs=305)
> print(f2t)

Factor Analysis using method =  minres
Call: fa(r = Harman23.cor$cov, nfactors = 2, rotate = "oblimin", n.obs = 305)
               item   MR1   MR2   h2   u2
height            1  0.87  0.08 0.84 0.16
arm.span          2  0.96 -0.05 0.89 0.11
forearm           3  0.93 -0.04 0.83 0.17
lower.leg         4  0.88  0.04 0.81 0.19
weight            5  0.01  0.94 0.89 0.11
bitro.diameter    6  0.00  0.80 0.64 0.36
chest.girth       7 -0.06  0.79 0.59 0.41
chest.width       8  0.13  0.62 0.47 0.53

                 MR1  MR2
SS loadings    3.37 2.58
Proportion Var 0.42 0.32
Cumulative Var 0.42 0.74

 With factor correlations of
     MR1  MR2
MR1 1.00 0.46
MR2 0.46 1.00

Test of the hypothesis that 2 factors are sufficient.

The degrees of freedom for the model is 13  and the objective function was  0.26
The number of observations was  305  with Chi Square =  76.38  with prob <  5.3e-11

Fit based upon off diagonal values = 1
Measures of factor score adequacy
                                                 [,1] [,2]
Correlation of scores with factors               0.98 0.96
Multiple R square of scores with factors         0.96 0.93
Minimum correlation of factor score estimates 0.91 0.85
```

Wiggins (1979) circumplex structures have been studied by those interested in the interpersonal dimensions of agency and communion Acton and Revelle (2002); Conte and Plutchik (1981); Gurtman (1993, 1997); Gurtman and Pincus (2003). A circumplex may be represented as coordinates in a two dimensional space, although some prefer to represent the items in *polar coordinates* (e.g., Gurtman (1997)). Mood and emotion data are typically thought of as representing two dimensions (energetic and tense arousal (Rafaeli and Revelle, 2006; Thayer, 1978, 1989) or alternatively Positive and Negative Affect (Larsen and Diener, 1992; Watson and Tellegen, 1985)). Tests for circumplex structure have been proposed by Acton and Revelle (2004) and Browne (1992). A subset of the Acton and Revelle (2004) tests have been implemented in the `circ.tests` function.

The appeal of circumplex structures to those who use them seems to be that there is no preferred set of axes and all rotations are equally interpretable. For those who look for underlying causal bases for psychometric structures, the circumplex solutions lack the simplicity found in simple structure. To some extent, the distinction between simple and circumplex structures is merely one of item selection. Measures of affect, for example, can be chosen that show simple structure, or with a more complete sampling of items, that show circumplex structure. The simulation functions `item.sim` and `circ.sim` have been develop to help understand the effects of item characteristics on structure.

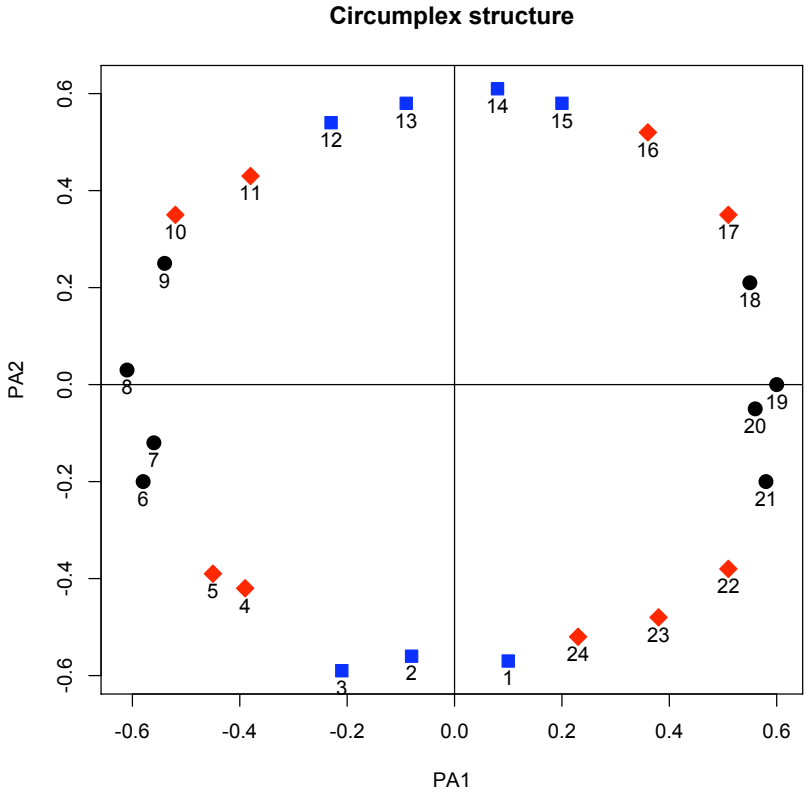

Fig. 6.6 A simulation of 24 items showing a circumplex structure. Simple structure would result if the items shown in diamonds were not included. Items generated using `circ.sim` and plotted using `cluster.plot`.

A mixture of a conventional five dimensional model of personality with a circumplex solution recognizes that most items are of complexity two and thus need to be represented in the 10 pairs of five dimensions Hofstee et al. (1992a). This solution (the "Abridged Big Five Circumplex" or "AB5C") recognizes that the primary loadings for items can be represented in a five dimensional space, and that items are loaded on at most two dimensions. One way of reporting factor structures that have a circumplex structure is to convert the loadings

into polar coordinates and organize the results by angle and radius length (see `polar`). Polar coordinate representations of emotion words allow for a clearer understanding of what words are near each other in the two dimensional space than does not the more conventional Cartesian representation (Rafaeli and Revelle, 2006).

6.3.4 Hierarchical and higher order models

If factors or components are allowed to correlate, there is implicitly a higher order factor that accounts for those correlations. This is the structure proposed for abilities (Carroll, 1993) where lower level specific abilities may be seen as correlating and representing the effect of a third stratum, general intelligence. An example of such a solution is discussed by Jensen and Weng (1994) in terms of how to extract a *g* factor of intelligence. The Schmid-Leiman transformation (Schmid and Leiman, 1957) that they apply is a constrained model of general higher order solutions (Yung et al., 1999). This is implemented in the `omega` and `schmid` functions in **psych**. An example applied to nine correlated variables is shown in Tables 6.13 and 6.14. Two alternative representations are seen in Figure 6.7. The first is a hierarchical solution with a higher order factor accounting for the correlation between the lower order factors. The second is solution in which the g factor identified in the hierarchical solution is then partialled out of the items and then the correlations of these residualized items is accounted for by three factors. These three lower level factors are orthogonal and represent pure group without the influence of the general factor.

It is important to understand how this analysis is done. The original orthogonal factor matrix (shown in Table 6.14 can be transformed obliquely using either the `oblimin` or `target.rot` transformation functions and then the correlations of those factors are themselves factored (See the top figure in Figure 6.7. g loadings of the items are then found from the product of the item loadings on the factors with the factor loadings on g. The resulting loadings on the orthogonal group factors are then found by subtracting the amount of variance for each item associated with g from the item communality. The square root of this is the loading on the orthogonalized group factors (Schmid and Leiman, 1957) (see the bottom half of Figure 6.7). This sort of *hierarchical* or *bifactor* solution is used to find the ω_h coefficient of general factor saturation (see 7.2.5).

Given the normal meaning of the word *hierarchy* to represent a tree like structure, there is an unfortunate naming confusion in these various models. While Gustafsson and Balke (1993), McDonald (1999) and Yung et al. (1999) refer to the structure in the top panel of Figure 6.7 as a *higher order* model and to solutions similar to those of the lower panel (Schmid and Leiman, 1957) as *hierarchical* models or *nested* models, Mulaik and Quartetti (1997) refers to the top panel as *hierarchical* and the lower panel as *nested*. Continuing the confusion, Chen et al. (2006) refer to the upper panel as a *second order* factoring, and the lower panel as a *bifactor* solution.

6.3.5 Comparing factor solutions

To evaluate the similarity between alternative factor or component solutions, it is possible to find the *factor congruence* coefficient by using the `factor.congruence` function. For a single

Table 6.13 An example hierarchical correlation matrix taken from Jensen and Weng (1994) with a bifactor solution demonstrating a Schmid-Leiman transformation and the extraction of a g factor. Loadings < .001 have been supressed.

```
> ability <- sim.hierarchical()           #create a hierarchical data structure
> round(ability,2)
> omega.ability <- omega(ability)   #solve it to show structure
> print(omega.ability,cut=.001)     #print the results, suppressing all values < .001
```

```
     V1   V2   V3   V4   V5   V6   V7   V8   V9
V1 1.00 0.56 0.48 0.40 0.35 0.29 0.30 0.25 0.20
V2 0.56 1.00 0.42 0.35 0.30 0.25 0.26 0.22 0.18
V3 0.48 0.42 1.00 0.30 0.26 0.22 0.23 0.19 0.15
V4 0.40 0.35 0.30 1.00 0.42 0.35 0.24 0.20 0.16
V5 0.35 0.30 0.26 0.42 1.00 0.30 0.20 0.17 0.13
V6 0.29 0.25 0.22 0.35 0.30 1.00 0.17 0.14 0.11
V7 0.30 0.26 0.23 0.24 0.20 0.17 1.00 0.30 0.24
V8 0.25 0.22 0.19 0.20 0.17 0.14 0.30 1.00 0.20
V9 0.20 0.18 0.15 0.16 0.13 0.11 0.24 0.20 1.00


Omega
Call: omega(m = ability)
Alpha:  0.76
G.6:   0.76
Omega Hierarchical:   0.69
Omega H asymptotic:   0.86
Omega Total   0.8

Schmid Leiman Factor loadings greater than  0.001
   g    F1*  F2*  F3*  h2   u2
V1 0.72 0.35           0.64 0.36
V2 0.63 0.31           0.49 0.51
V3 0.54 0.26           0.36 0.64
V4 0.56      0.42      0.49 0.51
V5 0.48      0.36      0.36 0.64
V6 0.40      0.30      0.25 0.75
V7 0.42           0.43 0.36 0.64
V8 0.35           0.36 0.25 0.75
V9 0.28           0.29 0.16 0.84

With eigenvalues of:
   g  F1*  F2*  F3*
2.29 0.29 0.40 0.40

general/max  5.74   max/min =   1.39
The degrees of freedom for the model is 12  and the fit was  0
```

Table 6.14 An orthogonal factor solution to the ability correlation matrix of Table 6.13 may be rotated obliquely using `oblimin` and then those correlations may be factored. g loadings of the items are then found from the product of the item loadings on the factors times the factor loadings on g. This procedure results in the solution seen in Table 6.13.

```
> print(omega.abilty,all=TRUE)  #show all the output from omega

$schmid$orthog

Loadings:
   F1   F2   F3
V1 0.70 0.30 0.25
V2 0.61 0.26 0.22
V3 0.52 0.22 0.19
V4 0.24 0.63 0.18
V5 0.21 0.54 0.16
V6 0.17 0.45 0.13
V7 0.17 0.15 0.56
V8 0.14 0.12 0.46
V9 0.11 0.10 0.37

                  F1    F2    F3
SS loadings    1.324 1.144 0.884
Proportion Var 0.147 0.127 0.098
Cumulative Var 0.147 0.274 0.372

$schmid$oblique
    F1  F2  F3
V1 0.8 0.0 0.0
V2 0.7 0.0 0.0
V3 0.6 0.0 0.0
V4 0.0 0.7 0.0
V5 0.0 0.6 0.0
V6 0.0 0.5 0.0
V7 0.0 0.0 0.6
V8 0.0 0.0 0.5
V9 0.0 0.0 0.4

$schmid$phi
     F1   F2   F3
F1 1.00 0.72 0.63
F2 0.72 1.00 0.56
F3 0.63 0.56 1.00

$schmid$gloading

Loadings:
   Factor1
F1 0.9
F2 0.8
F3 0.7

               Factor1
SS loadings      1.940
Proportion Var   0.647
```

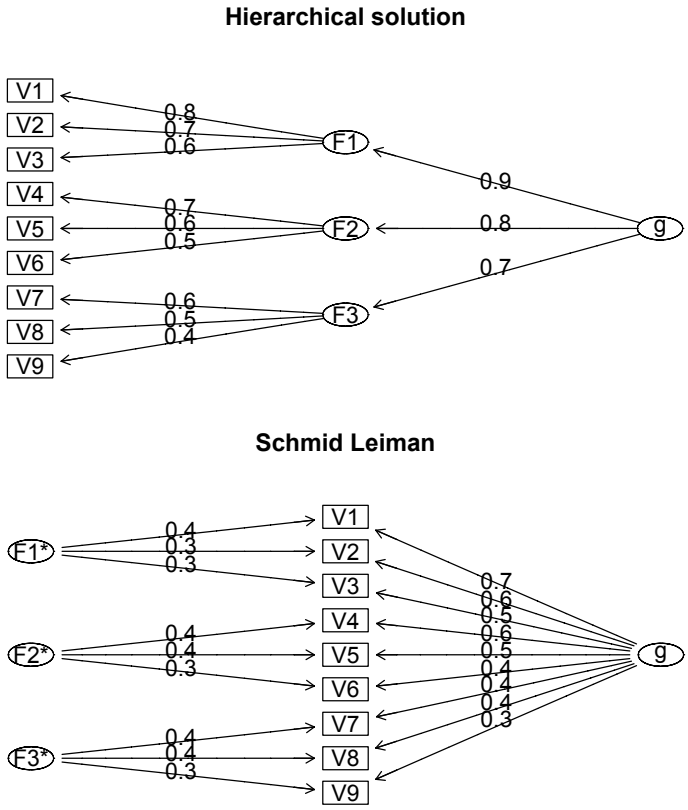

Fig. 6.7 The ω_h (omega-hierarchical) coefficient of reliability may be calculated from a hierarchical factor structure by using the Schmid Leiman transformation. The original factors are transformed to an oblique solution (top panel) and then the general factor is extracted from the items. The remaining factors (F1*, F2*, and F3*) are now orthogonal and reflect that part of the item that does not have g in it. Figures drawn using the `omega` function. The correlation matrix is shown in Table 6.13

pair of factors this is simply

$$r_c = \frac{\sum_1^n F_{xi} F_{yi}}{\sqrt{\sum_1^n F_{xi}^2 \sum_1^n F_{yi}^2}}. \tag{6.19}$$

The *congruence coefficient* is the cosine of the angle of the two factor loading vectors, taken from the origin. Compare this to Equation 4.22. The correlation is the cosine of the angle of the two loadings vectors, taken from the mean loading on each vector. Generalizing Equation 6.19 to the matrix of congruence coefficients, and expressing it in matrix algebra, the congruence coefficient $\mathbf{R}_c$ for two matrices of factor loadings $\mathbf{F}_x$ and $\mathbf{F}_y$ is

$$\mathbf{R_c} = diag(\mathbf{F_x F_x}')^{-1/2} \mathbf{F_x F_y}' diag(\mathbf{F_y F_y}')^{-1/2} \tag{6.20}$$

where

$$diag(\mathbf{F_x F_x}')^{-1/2}$$

is just dividing by the square root of the sum of squared loadings. Although similar in form to a correlation coefficient (Equation 4.24), the difference is that the mean loading is not subtracted when doing the calculations. Thus, even though two patterns are highly correlated, if they differ in the mean loading, they will not have a high factor congruence. Consider the factor congruence and correlations between the factors found in Tables 6.8 and 6.9.

```
> round(factor.congruence(f2,pc2),3)
```

```
          PC1   PC2
Factor1 0.998 0.000
Factor2 0.000 0.997
```

```
> round(cor(f2$loadings,pc2$loadings),3)
```

```
           PC1    PC2
Factor1  0.997 -0.981
Factor2 -0.969  0.993
```

Although the cross component loading congruences are 0.0, they are highly negatively correlated (-.97 and -.98). The negative correlations are reflecting that the patterns of loadings are high versus 0 and Factor or Component 1, and 0 versus high on Factor or Component 2. By subtracting the mean loading, as is done in the correlation, this makes the cross components (1 with 2 and 2 with 1) highly negatively correlated even they they are not congruent.

6.4 The number of factors/components problem

A fundamental question in both components and factor analysis is how many components or factors to extract? While it is clear that more factors will fit better than fewer factors, and that more components will always summarize the data better than fewer such an improvement in fit has a cost in parsimony. Henry Kaiser once said that "a solution to the number-of factors problem in factor analysis is easy", that he used to make up one every morning before breakfast. But the problem, of course is to find *the* solution, or at least a solution that others will regard quite highly not as the best" (Horn and Engstrom, 1979). There are at least eight procedures that have been suggested:

1) Extracting factors until the χ^2 of the residual matrix is not significant. Although statistically this makes sense, it is very sensitive to the number of subjects in the data set. That is, the more subjects analyzed, the more factors or components that will be extracted. The rule is also sensitive to departures from normality in the data as well as the assumption that residual error is random rather than systematic but small. χ^2 estimates are reported for the maximum likelihood solution done by the `factanal` function and all solutions using `fa`.

2) Extracting factors until the change in χ^2 from factor n to factor n+1 is not significant. The same arguments apply to this rule as the previous rule. That is, increases in sample size will lead to an increase in the number of factors extracted.

3) Extracting factors until the eigenvalues of the real data are less than the corresponding eigenvalues of a random data set of the same size (*parallel analysis*)(Dinno, 2009; Hayton et al., 2004; Horn, 1965; Humphreys and Montanelli, 1975; Montanelli and Humphreys, 1976). The `fa.parallel` function plots the eigenvalues for a principal components solution as well as the eigen values when the communalities are estimated by a one factor *minres* solution

for a given data set as well as that of n (default value = 20) randomly generated parallel data sets of the same number of variables and subjects. In addition, if the original data are available, 20 artificial data sets are formed by resampling (with replacement) from these data (Figure 6.8). A typical application is shown for 24 mental ability tests discussed by Harman (1960, 1976) and reported originally by Holzinger and Swineford (1939) and available as the `Harman74.cor` data set. Parallel analysis is partially sensitive to sample size in that for large samples the eigenvalues of random factors will tend to be very small and thus the number of components or factors will tend to be more than other rules. Although other parallel analysis functions use *SMC*s as estimates of the communalties (e.g., the `paran` in the **paran** package), simulations using the `sim.parallel` function suggest that the `fa.parallel` solution is a more accurate estimate of the number of major factors in the presence of many small, minor factors.

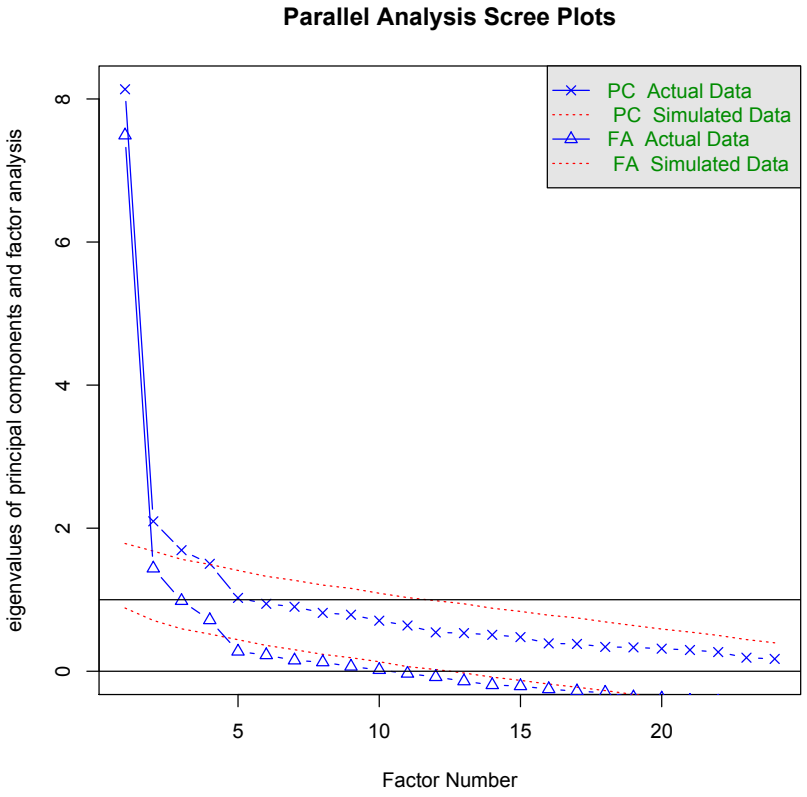

Fig. 6.8 A parallel analysis of 24 mental ability tests is found by the `fa.parallel` function. The eigenvalues of the principal components solution of 20 random data sets suggests four components. A similar solution is found from the parallel factors. When the raw data are available, eigen values for the correlation matrices formed from random samples (with replacement) of the raw data are also examined.

4) Plotting the magnitude of the successive eigenvalues and applying the *scree test* (a sudden drop in eigenvalues analogous to the change in slope seen when scrambling up the talus slope of a mountain and approaching the rock face) (Cattell, 1966c). In the example of the 24 mental tests case of Harman-Holzinger-Swineford (Figure 6.8), a strong argument could be made for either one factor or four factors.

5) Extracting factors as long as they are interpretable. A surprisingly compelling rule for the number of factors or components to extract. This basically reflects common sense. The disadvantage is that investigators differ in their ability or desire to interpret factors. While some will find a two factor solution most interpretable, others will prefer five.

6) Using the *Very Simple Structure* Criterion (`VSS`), (Revelle and Rocklin, 1979) (Figure 6.9). Most people, when interpreting a factor solution, will highlight the large (salient) loadings and ignore the small loadings. That is, they are interpreting the factor matrix as if it had simple structure. How well does this simplified matrix reproduce the original matrix. That is, if c is the complexity (number of non zero loadings) of an item and max_c means the greatest (absolute) c loadings for an item, then find the *Very Simple Structure* matrix, $\mathbf{S_c}$, where

$$s_{cij} = \begin{pmatrix} f_{ij} \; if(f_{ij} = max_c(fi.)) \\ 0 \qquad otherwise \end{pmatrix}$$

Then let

$$\mathbf{R}^*_{sc} = \mathbf{R} - \mathbf{SS}' \tag{6.21}$$

and

$$VSS_c = 1 - \frac{\mathbf{1R^*_{sc}}^2\mathbf{1}' - diag(\mathbf{R^*_{sc}})}{\mathbf{1R}^2\mathbf{1}' - diag(\mathbf{R})}$$

That is, the VSS criterion for a complexity c is 1 - the squared residual correlations divided by the squared observed correlations, where the residuals are found by the "simplified" factor equation 6.21. Example of two such VSS solutions, one for the Holzinger-Harmon problem of 24 mental tests and one for a set of 24 simulated variables representing a *circumplex* structure in two dimensions are seen in Figure 6.9. The presence of a large general factor in the Holzinger data set leads to an identification of one factor as being optimal if the tests are seen as having complexity one, but four factors if the tests are of complexity two or three. Compare this result to those suggested by the *scree* test or the *parallel factors* tests seen in Figure 6.8. Unlike $\mathbf{R}^*$ as found in equation 6.5, $\mathbf{R}^*_{sc}$ as found in equation 6.21 is sensitive to rotation and can also be used for evaluating alternative rotations. The *Very Simple Structure* criterion is implemented in the `VSS` and `VSS.plot` functions in the **psych** package.

7) Using the Minimum Average Partial criterion (MAP). Velicer (1976) proposed that the appropriate number of components to extract is that which minimizes the average (squared) partial correlation. Considering the $(n+p)x(n+p)$ super matrix composed of the *nxn* correlation matrix $\mathbf{R}$, the *nxp* component matrix $\mathbf{C}$, and the *pxp* covariance matrix of the components $\mathbf{CC}'$

$$\begin{pmatrix} \mathbf{R} & \mathbf{C}' \\ \mathbf{C} & \mathbf{CC}' \end{pmatrix} = \begin{pmatrix} \mathbf{R} & \mathbf{C}' \\ \mathbf{C} & \mathbf{I} \end{pmatrix}$$

then, partialling the components out of the correlation matrix produces a matrix of partial covariances, $\mathbf{R}^* = \mathbf{R} - \mathbf{CC}'$ and the partial correlation matrix $\mathbf{R}^\#$ is found by dividing the partial covariances by their respective partial variances

$$\mathbf{R}^\# = \mathbf{D}^{-1/2}\mathbf{R}^*\mathbf{D}^{-1/2} \tag{6.22}$$

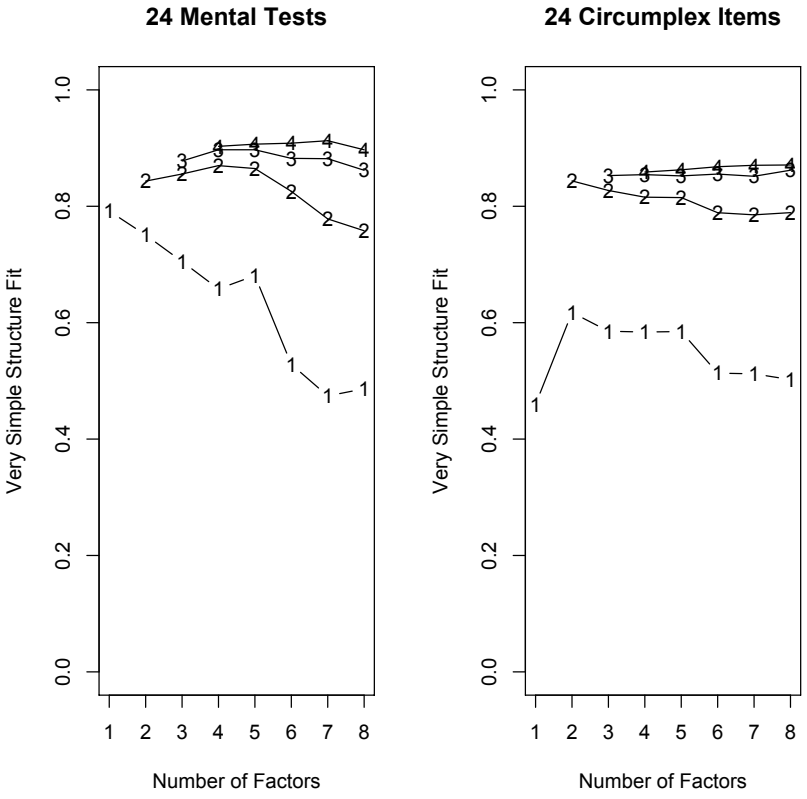

Fig. 6.9 Very Simple Structure for 24 mental ability tests (left panel) and 24 circumplex structured data (right panel). For the 24 mental tests, the complexity 1 solution suggests a large general factor but for complexity two or three, the best solution seems to be four factors. For the 24 circumplex items, the complexity one solution indicates that two factors are optimal, and the difference between the complexity one and two factor solutions suggests that the items are not simple structured (which, indeed, they are not.)

where $\mathbf{D} = diag(\mathbf{R}^*)$. The MAP criterion is just the sum of squared off diagonal elements of $\mathbf{R}^{\#}$. The logic of the MAP criterion is that although the residual correlations in $\mathbf{R}^*$ will become smaller as more factors are extracted, so will residual variances ($diag(\mathbf{R}^*)$). The magnitude of the partial correlations will thus diminish and then increase again when too many components are extracted. `MAP` is implemented in **psych** as part of the `VSS` function. For the Harman data set, MAP is at a minimum at four components, for the two dimensional circumplex strucure, at two:

```
> round(harman.vss$map,4)

[1] 0.0245 0.0216 0.0175 0.0174 0.0208 0.0237 0.0278 0.0301

> round(circ.vss$map,3)

[1] 0.044 0.006 0.008 0.011 0.014 0.017 0.020 0.024
```

This result agrees with VSS solution for these two problems.

8) Extracting principal components until the eigenvalue <1 (Kaiser, 1970). This is probably the most used and most disparaged rule for the number of components. The logic behind it is that a component should be at least as large as a single variable. Unfortunately, in practice the $\lambda > 1$ rule seems to be a very robust estimate of the number of variables divided by three (Revelle and Rocklin, 1979; Velicer and Jackson, 1990b).

Each of the procedures has its advantages and disadvantages. Using either the χ^2 test or the change in χ^2 test is, of course, sensitive to the number of subjects and leads to the nonsensical condition that if one wants to find many factors, one simlpy runs more subjects. The scree test is quite appealling but can lead to differences of interpretation as to when the scree "breaks". Extracting interpretable factors means that the number of factors reflects the investigators creativity more than the data. *VSS*, while very simple to understand, will not work very well if the data are very factorially complex. (Simulations suggests it will work fine if the complexities of some of the items are no more than 2). The eigenvalue of 1 rule, although the default for many programs, is fairly insensitive to the correct number and suggests that the number of factors is roughly 1/3 of the number of variables (Revelle and Rocklin, 1979; Velicer and Jackson, 1990b). It is the least recommended of the procedures (for recent reviews discussing how too many articles still use the eigen value of 1 rule, see Dinno (2009). MAP and VSS have the advantage that simulations show that they achieve a minimum (MAP) or maximum (VSS) at the correct number of components or factors (Revelle and Rocklin, 1979; Zwick and Velicer, 1982).

The number of factors problem was aptly summarized by Clyde Coombs who would say that determining the number of factors was like saying how many clouds were in the sky. On a clear day, it was easy, but when it was overcast, the problem became much more complicated. That the number of factors problem is important and that the standard default option in commercial packages such as SPSS is inappropriate has been frequently bemoaned (Dinno, 2009; Preacher and MacCallum, 2003) and remains a challenge to the interpretation of many factor analyses. Perhaps the one clear recommendation is to *not* use the eigen value of 1 rule. Looking for consistency between the results of *parallel analysis*, the *scree*, the *MAP* and *VSS* tests and then trying to understand why the results differ is probably the most recommended way to choose the number of factors.

6.5 The number of subjects problem

A recurring question in factor analysis and principal components analysis is how many subjects are needed to get a stable estimate? Although many rules of thumb have been proposed, suggesting minima of 250 or 500 and ratios as high as 10 times as many subjects as variables, the answer actually depends upon the clarity of the structure being examined (MacCallum et al., 1999, 2001). Most important is the communality of the variables. As the communality of the variables goes up, the number of observations necessary to give a clean structure goes down. A related concept is one of overdetermination. As the number of markers for a factor increase, the sample size needed decreases. That is, if the ratio of the number of variables to number of factors is high (e.g., 20:3), and the communalities are high (> .70), then sample sizes as small as 60-100 are quite adequate. But, if that ratio is low, then even with large sample sizes (N=400) , admissible solutions are not guaranteed. Although more subjects are always better, it is more important to have good markers for each factor (high communalities)

as well as many markers (a high variable to factor ratio) than it is to increase the number of subjects.

Unfortunately, although it is wonderful advice to have many markers of high communality (as is the case when examining the structure of tests), if using factor analysis to analyze the structure of items rather than tests, the communalilties will tend to be fairly low ($.2 < h^2 < .4$ is not uncommon). In this case, increasing the number of markers per factor, and increasing the number of subjects is advised. It is useful to examine the likelihood of achieving an acceptable solution as a function of low communalities, small samples, or the number of markers by simulation. Using the `sim.circ`, `sim.item` or `VSS.sim` functions, it is straightforward to generate samples of various sizes, communalities, and complexities.

6.6 The problem of factoring items

It is not uncommon to use factor analysis or related methods (see below) to analyze the structure of ability or personality items. Unfortunately, item difficulty or endorsement frequencies can seriously affect the conclusions. As discussed earlier (4.5.1.4), the ϕ correlation, which is a Pearson correlation for dichotomous items, is greatly affected by differences in item endorsement frequencies. This means that the correlations between items reflects both shared variance as well as differences in difficulty. Factoring the resulting correlation matrix can lead to a spurious factor reflecting differences in difficulty. Consider the correlations of 9 items all loadings of .7 on a factor, but differing in their endorsement frequencies from 4% to 96%. These items were generated using the `sim.npn` function discussed in Chapter 8. A *parallel analysis* of these correlations suggests that two factors are present. These two correlated factors represent the two ends of the difficulty spectrum with the first factor reflecting the first six items, the second the last three. The same data matrix yields more consistent *tetrachoric* correlations and parallel analysis suggests one factor is present. A graphic figure using `fa.diagram` of these two results is in Figure 6.10.

6.7 Confirmatory Factor Analysis

Exploratory factor analysis is typically conducted in the early stages of scale development. The questions addressed are how many constructs are being measured; how well are they measured, and what are the correlations between the constructs? Exploratory factor analysis is just that, exploratory. It is a very powerful tool for understanding the measures that one is using and should be done routinely when using a new measure. Statistical and psychometric goodness of fit tests are available that analyze how well a particular exploratory model accounts for the data.

But many investigators want to go beyond exploring their data and prefer to test how well a particular model, derived a priori, fits the data. This problem will be addressed more fully when we consider *structural equation modeling*, *SEM*, in chapter 10 which combine a *measurement model* with a *structural model* (or a validity model). Confirmatory factor analysis is done when evaluating the measurement model. The process is logically very simple: specify the exact loadings in the pattern matrix and the exact correlations in the factor intercorrelation matrix. Then test how well this model reproduces the data. This is rarely done,

Table 6.15 Nine items are simulated using the `sim.npn` funciton with endorsement frequencies ranging from 96% to 4%. Part A: The correlations of 9 items differing in item endorsement but all with equal saturation of a single factor. The number of factors suggested by `fa.parallel` is two. Part B: The correlations based upon the *tetrachoric* correlation are both higher and more similar. The number of factors suggested by `fa.parallel` is one.

```
Part A:
> set.seed(17)
> items <- sim.npn(9,1000,low=-2.5,high=2.5)$items
> describe(items)
```

```
  var    n mean   sd median trimmed mad min max range  skew kurtosis   se
1   1 1000 0.96 0.21      1    1.00   0   0   1     1 -4.44    17.73 0.01
2   2 1000 0.90 0.30      1    1.00   0   0   1     1 -2.72     5.40 0.01
3   3 1000 0.80 0.40      1    0.88   0   0   1     1 -1.51     0.27 0.01
4   4 1000 0.69 0.46      1    0.73   0   0   1     1 -0.80    -1.36 0.01
5   5 1000 0.51 0.50      1    0.51   0   0   1     1 -0.03    -2.00 0.02
6   6 1000 0.33 0.47      0    0.29   0   0   1     1  0.72    -1.49 0.01
7   7 1000 0.17 0.37      0    0.08   0   0   1     1  1.77     1.15 0.01
8   8 1000 0.10 0.29      0    0.00   0   0   1     1  2.74     5.51 0.01
9   9 1000 0.04 0.20      0    0.00   0   0   1     1  4.50    18.26 0.01
```

```
> round(cor(items),2)
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
 [1,] 1.00 0.29 0.22 0.22 0.20 0.14 0.10 0.07 0.05
 [2,] 0.29 1.00 0.24 0.24 0.24 0.19 0.14 0.10 0.07
 [3,] 0.22 0.24 1.00 0.30 0.30 0.27 0.20 0.14 0.11
 [4,] 0.22 0.24 0.30 1.00 0.35 0.28 0.23 0.17 0.13
 [5,] 0.20 0.24 0.30 0.35 1.00 0.32 0.28 0.25 0.15
 [6,] 0.14 0.19 0.27 0.28 0.32 1.00 0.28 0.26 0.21
 [7,] 0.10 0.14 0.20 0.23 0.28 0.28 1.00 0.31 0.22
 [8,] 0.07 0.10 0.14 0.17 0.25 0.26 0.31 1.00 0.32
 [9,] 0.05 0.07 0.11 0.13 0.15 0.21 0.22 0.32 1.00
```

```
Part B:
> tet <- tetrachoric(items)
```

```
Call: tetrachoric(x = items)
tetrachoric correlation
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
 [1,] 1.00 0.62 0.52 0.56 0.64 0.58 0.50 0.39 0.24
 [2,] 0.62 1.00 0.47 0.48 0.55 0.52 0.57 0.45 0.39
 [3,] 0.52 0.47 1.00 0.51 0.53 0.56 0.57 0.47 0.53
 [4,] 0.56 0.48 0.51 1.00 0.54 0.49 0.52 0.47 0.56
 [5,] 0.64 0.55 0.53 0.54 1.00 0.50 0.53 0.59 0.45
 [6,] 0.58 0.52 0.56 0.49 0.50 1.00 0.48 0.52 0.54
 [7,] 0.50 0.57 0.57 0.52 0.53 0.48 1.00 0.57 0.51
 [8,] 0.39 0.45 0.47 0.47 0.59 0.52 0.57 1.00 0.65
 [9,] 0.24 0.39 0.53 0.56 0.45 0.54 0.51 0.65 1.00
```

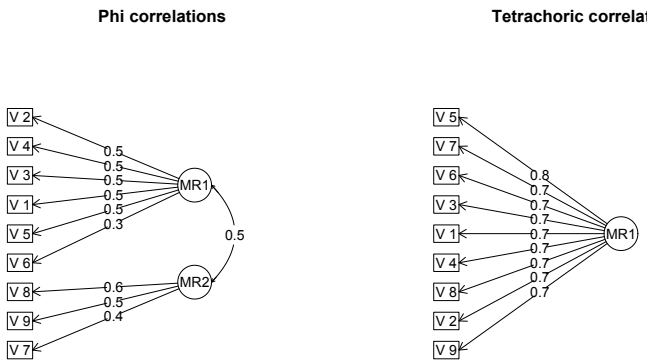

Fig. 6.10 Factor analyses of items can reflect difficulty factors unless the tetrachoric correlation is used. `fa.diagram(fa(items,2),main="Phi correlatons")` and `fa.diagram(fa(tet$rho),main="Tetrachoric correlations")`

but rather what is estimated is which loadings in a pattern matrix are non-zero and which correlations in the factor intercorrelation matrix are non-zero. The typical model estimation is done by a maximum likelihood algorithm (e.g. Jöreskog (1978)) in commercial packages such as EQS Bentler (1995), LISREL Jöreskog and Sörbom (1999) or Mplus Muthén and Muthén (2007) and open source programs such as Mx Neale (1994). SEM is implemented in R in the **sem** package written by John Fox (2006) as is OpenMx.

Consider a simple model that four tests (V1 ... V4) are all measures of the same construct, θ. The question to ask is then how well does each test measure θ? For this example, the four tests are created using the `sim.congeneric` function using population loadings on θ of .8, .7, .6, and .5. The input to `sem` specifies that the four variables have loadings to be estimated of a, b, c, and d. In addition, each variable has an error variance to be estimated (parameters u, v, w, and x). The variance of θ is arbitrarily set to be 1. These instructions are entered into the model matrix which is then given as input to `sem` (Table 6.16). That these results are very similar to what is observed with exploratory factor analysis is apparent when the *standardized loadings* from the `sem` output are compared to the results of `factanal` (Table 6.17).

The previous example tested the hypothesis that all four variables had significant loadings on the same factor. The results were identical to what was found using exploratory analysis. A much more compelling use of a confirmatory model is to test a more restricted hypotheses, e.g., that the loadings are the same. This is tested by revising the model to constrain the coefficients a,b, c, and d to be equal (Table 6.18). The χ^2 value shows that this model is not a good fit to the data. More importantly, the difference in χ^2 between the two models (.46 with df = 2 and 56.1 with df= 5) is also a χ^2 (55.5) with df = 3 or the difference in degrees of freedom between the two models. Thus the tau equivalent model may be rejected in favor of the (correct) congeneric model.

Table 6.16 Confirmatory factoring of four congeneric measures using the **sem** package. The data are simulated using the `sim.congeneric` function which returns the model as well as the observed correlation matrix and the simulated data if desired. The congeneric model allows the four factor loadings to differ. Compare this solution to the more constrained model of tau equivalence (Table 6.18).

```
>  set.seed(42)     #set the random number generator to allow for a comparison with other solutions
> congeneric <- sim.congeneric(N=1000,loads = c(.8,.7,.6,.5),short=FALSE)  #generate an artificial data set
> S <- cov(congeneric$observed)
>  model.congeneric <- matrix(c("theta ->  V1", "a", NA,
+  "theta -> V2", "b", NA,
+ "theta -> V3","c", NA,
+ "theta -> V4", "d", NA,
+  "V1 <-> V1", "u", NA,
+ "V2 <-> V2", "v", NA,
+ "V3 <-> V3", "w",  NA,
+ "V4 <-> V4", "x", NA,
+ "theta <-> theta", NA, 1), ncol = 3, byrow = TRUE)
>  model.congeneric          #show the model

      [,1]              [,2] [,3]
 [1,] "theta ->  V1"    "a"  NA
 [2,] "theta -> V2"     "b"  NA
 [3,] "theta -> V3"     "c"  NA
 [4,] "theta -> V4"     "d"  NA
 [5,] "V1 <-> V1"       "u"  NA
 [6,] "V2 <-> V2"       "v"  NA
 [7,] "V3 <-> V3"       "w"  NA
 [8,] "V4 <-> V4"       "x"  NA
 [9,] "theta <-> theta" NA   "1"

>  sem.congeneric = sem(model.congeneric,S, 1000)  #do the analysis
>  summary(sem.congeneric, digits = 3)

 Model Chisquare =  0.46   Df =  2 Pr(>Chisq) = 0.795
 Chisquare (null model) =  910   Df =  6
 Goodness-of-fit index =  1
 Adjusted goodness-of-fit index =  0.999
 RMSEA index =  0   90% CI: (NA, 0.0398)
 Bentler-Bonnett NFI =  1
 Tucker-Lewis NNFI =  1.01
 Bentler CFI =  1
 SRMR =  0.00415
 BIC =  -13.4
 Normalized Residuals
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-0.177000 -0.032200 -0.000271  0.010600  0.017000  0.319000
 Parameter Estimates
  Estimate Std Error z value Pr(>|z|)
a 0.829    0.0320    25.90   0        V1 <--- theta
b 0.657    0.0325    20.23   0        V2 <--- theta
c 0.632    0.0325    19.43   0        V3 <--- theta
d 0.503    0.0340    14.80   0        V4 <--- theta
u 0.316    0.0346     9.12   0        V1 <--> V1
v 0.580    0.0334    17.35   0        V2 <--> V2
w 0.604    0.0337    17.94   0        V3 <--> V3
x 0.776    0.0382    20.31   0        V4 <--> V4
 Iterations =  13
```

Table 6.17 The `sem` confirmatory factoring and `factanal` exploratory factoring yield identical results for four congeneric measures. This is as would be expected given that there is no difference in the parameters being estimated. Compare this to Table 6.18 where some parameters are constrained to be equal.

```
> std.coef(sem.congeneric)            #find the standardized path coefficients

    Std. Estimate
a a 0.82766       V1 <--- theta
b b 0.65302       V2 <--- theta
c c 0.63068       V3 <--- theta
d d 0.49584       V4 <--- theta

> factanal(congeneric,factors=1)


Call:
factanal(factors = 1, covmat = Cov.congeneric, n.obs = 1000)
Uniquenesses:
   V1    V2    V3    V4
0.315 0.574 0.602 0.754
Loadings:
   Factor1
V1 0.828
V2 0.653
V3 0.631
V4 0.496
               Factor1
SS loadings      1.755
Proportion Var   0.439
Test of the hypothesis that 1 factor is sufficient.
The chi square statistic is 0.46 on 2 degrees of freedom.
The p-value is 0.795
```

There are many goodness of fit statistics for SEM, partly in response to the problem that the χ^2 statistic is so powerful (Marsh et al., 1988, 2005). The meaning of these statistics as well as many more examples of confirmatory factor analysis will be considered in Chapter 10.

6.8 Alternative procedures for reducing the complexity of the data

In addition to factor analysis and principal components analysis, there are two more alternatives to the problem of reducing the complexity of the data. One approach, *multidimensional scaling* has been introduced before when considering how to represent groups of objects. MDS considers the correlation matrix as a set of (inverse) distances and attempts to fit these distances using a limited number of dimensions. The other alternative, *cluster analysis*, represents a large class of algorithms, some of which are particularly useful the practical problem of forming homogeneous subsets of items from a larger group of items. Although typically used for grouping subjects (objects), clustering techniques can also be applied to the grouping of items. As such, the resulting clusters are very similar to the results of components or factor analysis.

Table 6.18 Tau equivalence is a special case of congeneric tests in which factor loadings with the true score are all equal, but the variables have unequal error variances. This model can be tested by constraining the factor loadings to equality and then examining the fit statistics. Compare this result to the case where the parameters are free to vary in the complete congeneric case (Table 6.16).

```
>  model.allequal <- matrix(c("theta ->  V1", "a", NA,"theta -> V2", "a",NA,"theta -> V3","a",NA,
+  "theta -> V4", "a", NA,   "V1 <-> V1", "u", NA,   "V2 <-> V2", "v", NA,  "V3 <-> V3", "w", NA,
+   "V4 <-> V4", "x", NA,  "theta <-> theta", NA, 1), ncol = 3, byrow = TRUE)
>  sem.allequal = sem(model.allequal,S, 1000)  #do the analysis
>  summary(sem.allequal, digits = 3)

 Model Chisquare =  56.1   Df =  5 Pr(>Chisq) = 7.64e-11
 Chisquare (null model) =  910   Df =  6
 Goodness-of-fit index =  0.974
 Adjusted goodness-of-fit index =  0.947
 RMSEA index =  0.101   90% CI: (0.0783, 0.126)
 Bentler-Bonnett NFI =  0.938
 Tucker-Lewis NNFI =  0.932
 Bentler CFI =  0.943
 SRMR =  0.088
 BIC =  21.6

 Normalized Residuals
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -3.160  -2.890  -0.967  -0.418   2.290   3.000

 Parameter Estimates
  Estimate Std Error z value Pr(>|z|)
a 0.668    0.0202    33.2    0        V1 <--- theta
u 0.448    0.0270    16.6    0        V1 <--> V1
v 0.565    0.0315    18.0    0        V2 <--> V2
w 0.576    0.0319    18.1    0        V3 <--> V3
x 0.730    0.0386    18.9    0        V4 <--> V4
 Iterations =  10
```

6.8.1 MDS solutions remove the general factor

An alternative to factor analysis or components analysis is to analyze the correlation matrix by using *multiple dimensional scaling* (*MDS*). *Metric* and *non-metric MDS* techniques attempt to maximize the fit (minimize the residual, an index of which is known as *stress*) between a data matrix of distances and the distances calculated from a matrix of coordinates. This was discussed in 2.7 and 4.3.

When working with variables sharing a large general factor such as general intelligence in the cognitive domain, or neuroticism in the clinical domain, all the entries in the correlation matrix will be large and thus, the average correlation will be large. Especially if orthogonal rotations are used, it is difficult to see the structure of the smaller factors. A factor analytic solution is to use oblique transformations, extract second order factors, and then plot the loadings on the lower level factors. An alternative to this is to represent the correlations in terms of deviations from the average correlation. One way to do this is to convert the *correlations* to *distances* and do a *multidimensional scaling* of the distances. As was shown earlier (Equation 4.11) *distance* between two variables is an inverse function of their correlation:

$$d_{xy} = \sqrt{2 * (1 - r_{xy})}. \tag{6.23}$$

Applying formula 6.23 to the 24 ability measures of Holzinger-Swineford found in the `Harman74.cor` data set, results in a very interpretable two dimensional solution, with an implicit general dimension removed (Figure 6.11). The data points have been shown using different plotting symbols in order to show the cluster structure discussed below. Distances from the center of the figure represent lower correlations with the general factor. It appears as if the four factors of the factor analytic solution are represented in terms of the four quadrants of the MDS solution.

Table 6.19 Multidimensional scaling of 24 mental tests. See Figure 6.11 for a graphical representation. The correlations are transformed to distances using Equation 6.23. The multidimensional scaling uses the `cmdscale` function.

```
> dis24 <- sqrt(2*(1-Harman74.cor$cov))
> mds24 <- cmdscale(dis24,2)
> plot.char <- c( 19, 19, 19, 19,   21, 21,  21, 21,   21, 20, 20, 20,
    20, 23, 23, 23,   23, 23, 23, 19,   22, 19, 19, 22 )
> plot(mds24,xlim=c(-.6,.6),ylim=c(-.6,.6),xlab="Dimension 1",ylab="Dimension 2",asp=1,pch=plot.char)
> position <- c(2,2,3,4,  4,4,3,4,  3,2,3,2, 3,3,1,4, 4,1,3,1,  1,2,3,4)
>  text(mds24,rownames(mds24),cex=.6,pos=position)
> abline(v=0,h=0)
> title("Multidimensional Scaling of 24 ability tests")
> #draw circles at .25 and .50 units away from the center
>  segments = 51
>  angles <- (0:segments) * 2 * pi/segments
>         unit.circle <- cbind(cos(angles), sin(angles))
>    lines(unit.circle*.25)
>    lines(unit.circle*.5)
```

6.8.2 Cluster analysis – poor man's factor analysis?

Another alternative to factor or components analysis is *cluster analysis*. The goal of cluster analysis is the same as factor or components analysis (reduce the complexity of the data and attempt to identify homogeneous subgroupings). Mainly used for clustering people or objects (e.g., projectile points if an anthropologist, DNA if a biologist, galaxies if an astronomer), clustering may be used for clustering items or tests as well. Introduced to psychologists by Tryon (1939) in the 1930's, the cluster analytic literature exploded in the 1970s and 1980s (Blashfield, 1980; Blashfield and Aldenderfer, 1988; Everitt, 1974; Hartigan, 1975). Much of the research is in taxonmetric applications in biology Sneath and Sokal (1973); Sokal and Sneath (1963) and marketing (Cooksey and Soutar, 2006) where clustering remains very popular. It is also used for taxonomic work in forming clusters of people in family (Henry et al., 2005) and clinical psychology Martinent and Ferrand (2007); Mun et al. (2008). Interestingly enough it has has had limited applications to psychometrics. This is unfortunate, for as has been pointed out by e.g. Tryon (1935); Loevinger et al. (1953), the theory of factors, while mathematically compelling, offers little that the geneticist or behaviorist or perhaps

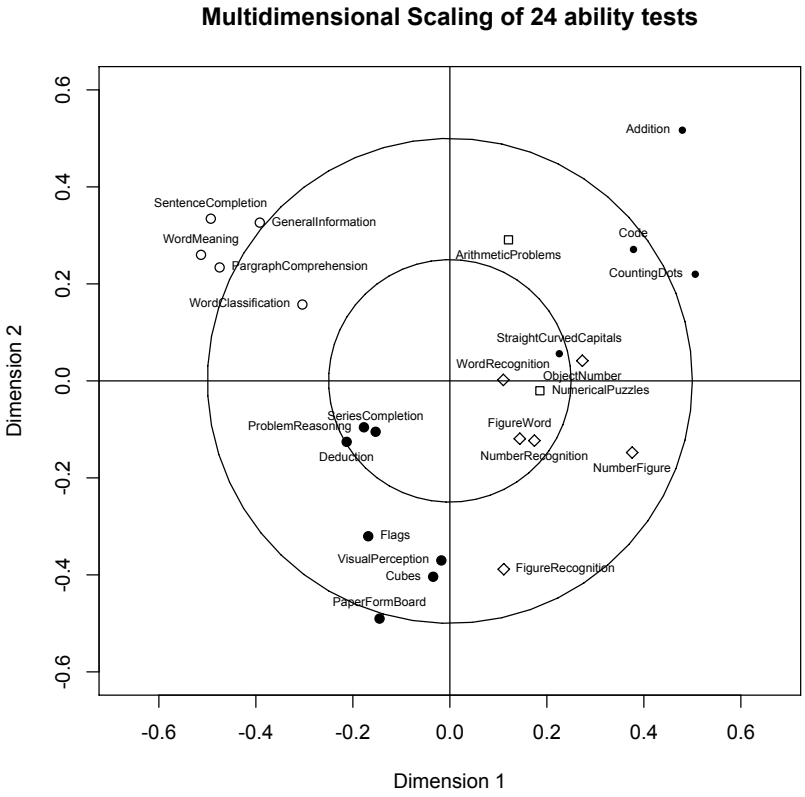

Fig. 6.11 A multidimensional scaling solution removes the general factor from 24 ability measurements. Distance from the center of the plot reflects distance from the general factor. Symbols reflect the hierarchical clusters groupings seen in Figure 6.13.

even non-specialist finds compelling. Cooksey and Soutar (2006) reviews why the ICLUST algorithm is particularly appropriate for scale construction in marketing.

As reviewed by Aldenderfer and Blashfield (1984); Blashfield (1980); Blashfield and Aldenderfer (1988), there have been three main psychological approaches to cluster analysis. Tryon (1939) and Loevinger et al. (1953), working in the psychometric tradition of factor analysis, developed clustering algorithms as alternatives to factor analysis. Ward (1963), introduced a goodness of fit measure for hierarchical clustering based upon an analysis of variance perspective. This was adapted by Johnson (1967), who was influenced by the multidimensional scaling considerations of ordinal properties of distances (Kruskal, 1964) and considered hierarchical clustering from a non-metric perspective.

At the most abstract, clusters are just ways of partitioning the data or the proximity matrix. A clustering rule is merely a way of assigning n items to c clusters. For n= the number of items or tests, and c=number of clusters, the goal is to assign each item to one cluster. Thus, for a n item vector $\mathbf{g}$, assign a cluster number, $1 \leq c_g \leq c$, to each g_i. In terms of scoring the data, this may be conceptualizes as finding a n * c matrix of *keys*, $\mathbf{K}$, where

each k_{ij} is 1, 0, or -1, that optimally divides the matrix. Two items are said to be in the same cluster if they both have non-zero entries for that column of the **K** matrix. The keys matrix, when multiplied by the N * n data matrix, **X**, will produce a scores matrix, $\mathbf{S_{xk}}$, of dimension N * c. This scores matrix is merely a set of scores on each of the c scales and represents the sum of the items that define each scale. Such a way of scoring (just adding up the salient items) is, in fact, what is typically done for factor or component analysis. The problem then is how to find the grouping or clustering matrix, **K**. That is, what is the criterion of optimality? As pointed out by Loevinger et al. (1953), unlike factor analysis which weights items by their factor loadings, cluster analysis solves the practical problem of assigning items to tests on a all or none basis.

There are two limits for the **K** matrix. One is just the identity matrix of rank n. That is, no items are in the same cluster. The other is a one column matrix of all -1s or 1s. That is, all items are in one cluster. The problem is thus to determine what is the optimal number of clusters, c, and how to allocate items to these clusters. Optimality may be defined in multiple ways, and different clustering algorithms attempt to maximize these different criteria. Loevinger et al. (1953) tried to maximize the within cluster homogeneity in terms similar to those of reliability theory. Ward (1963) evaluated clusters by a within cluster versus between cluster reduction of variance estimate.

There are two broad classes of clustering algorithms: *hierarchical* and *non-hierarchical*. Within hierarchical, there are also two broad classes: *divisive* and *agglomerative*. Finally, there is the distinction between *overlapping* and *non-overlapping clusters*. That is, can an item be in more than one cluster. Some of these distinctions will make more sense if we consider two examples, one, developed by Loevinger et al. (1953) is non-hierarchial, the other, implemented in the `ICLUST` algorithm is an agglomerative, hierarchical algorithm that allocates items to distinct clusters and that was specifically developed for the problem of forming homogeneous clusters of items (Revelle, 1979).

6.8.2.1 Non-hierarchical clustering

The process of forming clusters non-hierarchically was introduced for scale construction by Loevinger et al. (1953). The steps in that procedure are very similar to those found in other non-hierarchical algorithms.

1. Find the proximity (e.g. correlation) matrix,
2. Identify the most similar triplet of items,
3. Combine this most similar triplet of items to form a seed cluster,
4. Add items to this cluster that increase the cluster saturation,
5. Eliminate items that would reduce the cluster saturation,
6. Repeat steps 4 and 5 until no items remain. This is a homogeneous subtest.
7. Identify a new nucleus set of three items from the remaining items and start at step 3.
8. Repeat steps 4-7 until no items remain. The clusters are both maximally homogeneous and maximally independent.

Step 1, finding a proximity matrix, requires a consideration of what is meant by being similar. Typically, when constructing tests by combining subsets of items, an appropriate index of similarity is the correlation coefficient. Alternatives include covariances, or perhaps some inverse function of distance. A question arises as to whether to reverse score items that are negatively correlated with the cluster. Although this makes good sense for items (“I do

not feel anxious" when reversed scored correlates highly with "other people think of me as very anxious"), it is problematic when clustering objects (e.g., people). For if two people are very dissimilar, does that necessarily mean that the negative of one person is similar to the other?

Steps 2 and 3, identifying the most similar triplet, requires both a theoretical examination of the content as well as identifying that triplet that has the highest covariances with each other. This set is then seen as the nucleus of the subtest to be developed. With suitably created items, it is likely that the nucleus will be easily defined. The variance of this cluster is just the sum of the item covariances and variances of the three items. Its covariance with other items is just the sum of the separate covariances.

Steps 4 and 5, add items to maximize the *cluster saturation* and reject items that would reduce it. Cluster saturation is merely the ratio of the sum of the item covariances or correlations divided by the total test variance:

$$S = \frac{\mathbf{1'V1} - diag(\mathbf{V})}{\mathbf{1'V1}}.$$

For an n item test, saturation is a function of α (see 7.2.3):

$$\alpha = S\frac{n}{n-1}.$$

To prevent "functional drift" as the clusters are formed, items once rejected from a cluster are not considered for that cluster at later stages.

Steps 4 and 5 are continued until S fails to increase.

Step 7 may be done sequentially, or, alternatively, multiple seed clusters could have been identified and the process would proceed in parallel.

If the goal is to efficiently cluster a group of items around a specified set of k seeds, or a randomly chosen set of k seeds, the `kmeans` function implements the Hartigan and Wong (1979) algorithm. This does not provide the direct information necessary for evaluating the psychometric properties of the resulting scales, but with a few steps, the cluster output can be converted to the appropriate keys matrix and the items can be scored. That is, `cluster2keys` takes the cluster output of a `kmeans` clustering and converts it to a keys matrix suitable for `score.items` or `cluster.cor`. Applying the `kmeans` function to the Harman data set, produces the cluster membership seen in Figure 6.11.

Although the `kmeans` algorithm works very well if the items are all positively correlated, the solution when items need to be reversed is not very good. This may be seen when forming clusters from items generated to represent both positive and negative loadings in a two dimensional space (Table 6.20). The data were generated using the `sim.item` function which defaults to producing items with two simple structured factors with high positive and negative loadings. When just two clusters were extracted, the solution was very power (Clusters 2.1 and 2.2 are more correlated with each other than they are internally consistent.) When four clusters are extracted they show the appropriate structure, with two sets of negatively correlated clusters (e.g., C4.1, C4.3 and C4.2, C4.4) independent of each other. Compare this solution to that shown in Table 6.21 where items were allowed to be reversed and a hierarchical clustering algorithm was used. This difference is most obvious when plotting the two cluster solutions from `kmeans` and `ICLUST` for 24 items representing a *circumplex* (see Figure 6.12).

Table 6.20 `kmeans` of simulated two dimensional data with positive and negative loadings produces two clusters that are highly negatively correlated and not very reliable (C2.1, C2.2). When four clusters are extracted (C4.1 ... C4.4), the resulting keys produce two pairs of negatively correlated clusters, with the pairs orthogonal to each other.

```
> set.seed(123)
> s2 <- sim.item(12)
> r2 <- cor(s2)
> k2 <- kmeans(r2,2)

> print(k2,digits=2)
K-means clustering with 2 clusters of sizes 6, 6
Cluster means:
   [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8]  [,9] [,10] [,11] [,12]
1 -0.16 -0.22 -0.22 -0.18 -0.17 -0.21  0.27  0.31  0.28  0.29  0.29  0.29
2  0.27  0.29  0.33  0.30  0.29  0.32 -0.18 -0.19 -0.19 -0.17 -0.21 -0.21
Clustering vector:
 [1] 2 2 2 2 2 2 1 1 1 1 1 1
Within cluster sum of squares by cluster:
[1] 6.0 5.9
Available components:
[1] "cluster"  "centers"  "withinss" "size"

> keys2 <- cluster2keys(k2)
> k4 <- kmeans(r2,4)
> keys4 <- cluster2keys(k4)
> keys24 <- cbind(keys2,keys4)
> colnames(keys24) <- c("C2.1","C2.2","C4.1","C4.2","C4.3", "C4.4")
> cluster.cor(keys24,r2)

Call: cluster.cor(keys = keys24, r.mat = r2)

(Standardized) Alpha:
C2.1 C2.2 C4.1 C4.2 C4.3 C4.4
0.50 0.54 0.64 0.62 0.64 0.66

(Standardized) G6*:
C2.1 C2.2 C4.1 C4.2 C4.3 C4.4
0.59 0.62 0.59 0.58 0.60 0.61

Average item correlation:
C2.1 C2.2 C4.1 C4.2 C4.3 C4.4
0.14 0.16 0.37 0.36 0.37 0.39

Number of items:
C2.1 C2.2 C4.1 C4.2 C4.3 C4.4
   6    6    3    3    3    3

Scale intercorrelations corrected for attenuation
 raw correlations below the diagonal, alpha on the diagonal
 corrected correlations above the diagonal:
      C2.1  C2.2  C4.1  C4.2  C4.3  C4.4
C2.1  0.50 -1.25  1.25  1.25 -0.85 -0.78
C2.2 -0.65  0.54 -0.80 -0.78  1.22  1.21
C4.1  0.71 -0.47  0.64  0.00 -0.03 -1.00
C4.2  0.70 -0.45  0.00  0.62 -1.05  0.02
C4.3 -0.48  0.71 -0.02 -0.66  0.64  0.04
C4.4 -0.45  0.72 -0.65  0.01  0.03  0.66
```

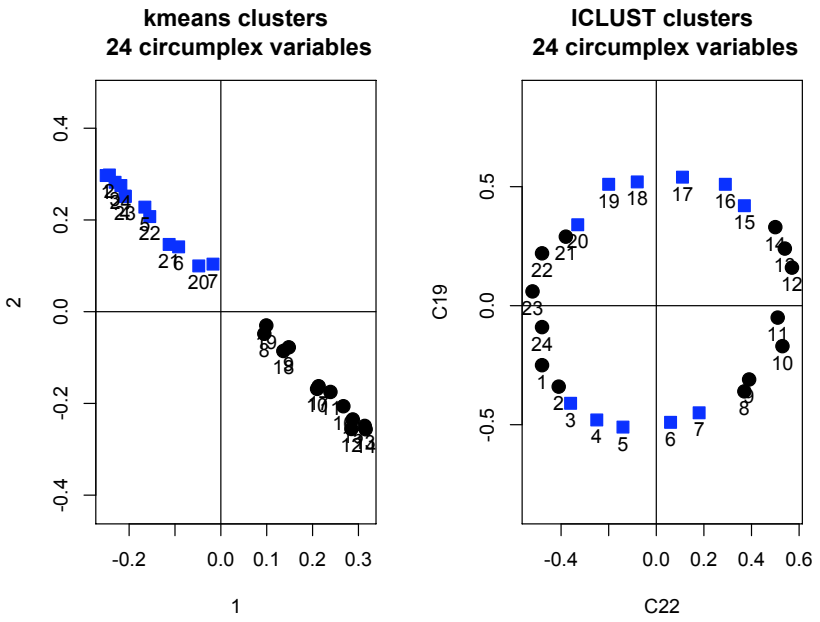

Fig. 6.12 When clustering items, it is appropriate to reverse key some items. When forming two clusters from data with a circumplex structure, the kmeans algorithm, which does not reverse key items, forms two highly negatively correlated clusters (left hand panel) while the ICLUST algorithm, which does reverse items, captures the two dimensional structure of the data (right hand panel).

6.8.2.2 Hierarchical clustering

Hierarchical cluster analysis forms clusters that are nested within clusters. The resulting *tree diagram* (also known somewhat pretentiously as a *rooted dendritic structure*) shows the nesting structure. Although there are many hierarchical clustering algorithms (e.g., `agnes`, `hclust`, and `ICLUST`), the one most applicable to the problems of scale construction is `ICLUST` (Revelle, 1979).

1. Find the proximity (e.g. correlation) matrix,
2. Identify the most similar pair of items
3. Combine this most similar pair of items to form a new variable (cluster),
4. Find the similarity of this cluster to all other items and clusters,
5. Repeat steps 2 and 3 until some criterion is reached (e.g., typicallly, if only one cluster remains or in `ICLUST` if there is a failure to increase reliability coefficients α or β).
6. Purify the solution by reassigning items to the most similar cluster center.

Just as for non-hierarchical problems, Step 1 requires a consideration of what is desired and will probably mean finding a correlation matrix.

Step 2, identifying the most similar pair might include adjusting the proximity matrix to correct for reliability of the measures. As discussed in Step 1 of the non-hierarchical question of whether to reverse key an item when considering similarity is very important. When

clustering items, it is probably a good idea to allow items to be reversed keyed. But when clustering objects, this is not as obvious.

Step 3, combining items i and j into a new cluster means adjusting the K matrix by forming a new column,c_{ij}, made up of the elements of c_i and c_j and then deleting these two columns.

Step 4, finding the similarity of the cluster with the remaining items can be done in many different ways. Johnson (1967), took a non-metric approach and considered the distance between two clusters to be defined by the *nearest neighbor* (also known as *single linkage*), the *furthest neighbor* (*maximum distance* or *complete linkage*). Metric techniques include considering the *centroid distance* or the *correlation*. Each of these choices may lead to different solutions. *single linkage* will lead to chaining, *complete linkage* will lead to very compact clusters.

Step 5, repeating steps 2 and 3 until some criterion is reached, is the essence of hierarchical clustering. Rather than just adding single items into clusters, hierarchical clustering allows for clusters to form higher level clusters. The important question is whether it is a good idea to combine clusters. Psychometric considerations suggest that clusters should be combined a long as the internal consistency of the cluster increases. There are at least three measures of internal consistency worth considering: the average inter-item correlation, the percentage of cluster variance that is reliable (e.g., coefficient α, (Cronbach, 1951)) and the percentage of cluster variance associated with one common factor (e.g., coefficient β, (Revelle, 1979; Zinbarg et al., 2005)). While the average r will not increase as higher order clusters are formed, both α and β typically will. However, α will increase even if the clusters are barely related, while β is more sensitive to over clustering and will not increase as frequently. To the extent that the cluster is meant to represent one construct, the β criterion seems more justified.

Step 6, purifying the clusters, is done only in the case that more than one cluster is identified. It is sometimes the case that hierarchically formed clusters will include items that are more similar to other clusters. A simple one or two stage purification step of reassigning items to the most similar cluster will frequently remove this problem.

It is interesting to compare the `ICLUST` cluster solution to the *Harman-Holzinger-Swinford* 24 mental measures (Figure 6.13) to the `cmdscale` multidimensional scaling solution in Figure 6.11. Cluster C20 corresponds to the lower right quadrant of the MDS, C17 to the lower and C10 to the upper region of the lower left quadrant (which then combine into C18), and C13 to the upper left quadrant. C15 groups the variables in the upper region of the upper right quadrant.

Cluster analysis results bear a great deal of similarity to early factor analytic work (e.g., Holzinger (1944)) which used various shortcuts to form groups of variables that were of approximately rank one within groups (clusters), but had some correlations between groups. The item by cluster loadings matrix is similar to a *factor structure* matrix and thus may be converted to a *cluster pattern matrix* by multiplying by the inverse of the cluster correlations. At least for the Harman 24 mental ability measures, it is interesting to compare the cluster pattern matrix with the oblique rotation solution from a factor analysis. The factor congruence of a four factor oblique pattern solution with the four cluster solution is $> .99$ for three of the four clusters and $> .97$ for the fourth cluster.

Table 6.21 Applying the ICLUST algorithm to the circumplex data set of Table 6.20 produces two orthogonal clusters with adequate reliability.

```
> ic <- ICLUST(r2)
> ic
> cluster.plot(ic)


ICLUST (Item Cluster Analysis)
Call: ICLUST(r.mat = r2)

Purified Alpha:
 C10   C9
0.79 0.78

G6* reliability:
 C10   C9
0.76 0.76

Original Beta:
 C10   C9
0.74 0.74

Cluster size:
C10  C9
  6   6

Item by Cluster Structure matrix:
      C10   C9
 [1,]  0.10 -0.63
 [2,] -0.05 -0.60
 [3,] -0.12 -0.59
 [4,] -0.59 -0.01
 [5,] -0.58  0.02
 [6,] -0.66 -0.03
 [7,] -0.04  0.59
 [8,]  0.06  0.57
 [9,] -0.03  0.62
[10,]  0.56 -0.01
[11,]  0.60  0.02
[12,]  0.62  0.02

With eigenvalues of:
C10  C9
2.2 2.2

Purified scale intercorrelations
 reliabilities on diagonal
 correlations corrected for attenuation above diagonal:
     C10   C9
C10 0.79 0.01
C9  0.01 0.78
```

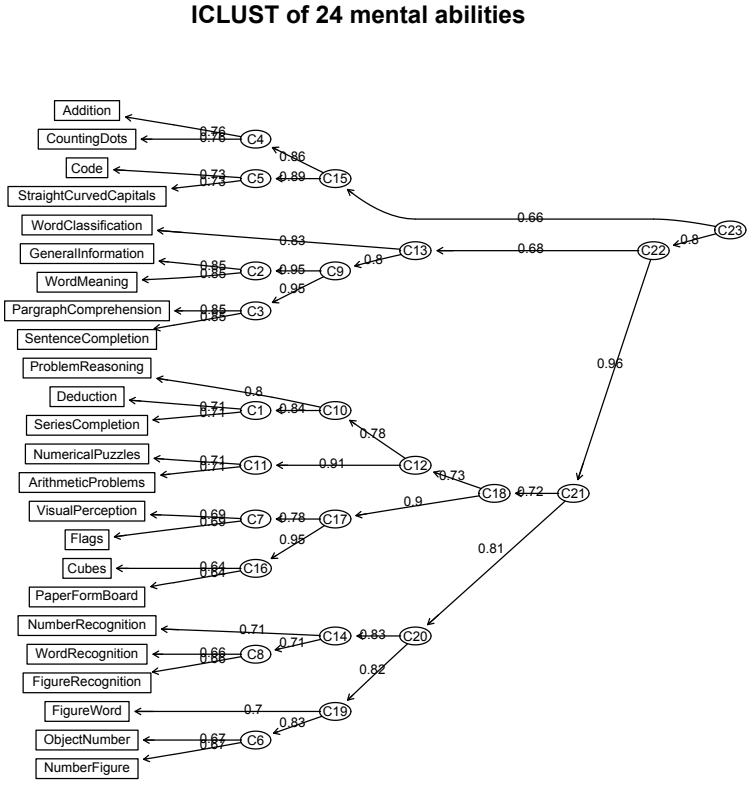

Fig. 6.13 The `ICLUST` hierarchical cluster analysis algorithm applied to 24 mental tests. The cluster structure shows a clear one cluster (factor) solution, with four clear subclusters (C13, C15, C18, and C 20). Graph created by `ICLUST`(Harman74.cor$cov). Compare this solution to that show in Figure 6.11.

Table 6.22 For the *Harman 24 mental measurements* the *oblique pattern matrix* from a factor analysis and the *pattern matrix* from ICLUST are practically identical as judged by their *factor congruence*.

```
> clust24 <- ICLUST(Harman74.cor$cov,4) #cluster the 24 mental abilities
> clust.pattern <- clust24$pattern
> f24 <- factanal(covmat=Harman74.cor,factors=4,rotation="oblimin")
> round(factor.congruence(clust.pattern,ob24),2)
```

```
    Factor1 Factor2 Factor3 Factor4
C18    0.30    0.97    0.07    0.26
C13    0.98    0.05    0.01    0.08
C20    0.19    0.25    0.21    0.99
C15    0.09    0.21    0.99    0.22
```

6.9 Estimating factor scores, finding component and cluster scores

Factors are latent variables representing the structural relationships between observed variables. Although we can not directly measure the factors, we can observe the correlations between the f factors and the v observed variables. This correlation matrix is known as the *pattern matrix*, ${}_v\mathbf{P}_f$. More typically interpreted, particularly with an oblique solution, is the *structure matrix*, ${}_v\mathbf{F}_f$, of beta weights for predicting the variables from the factors. The resulting factor loadings, communalities, and uniqueness are all defined by the model. As long as the number of free parameters (loadings) that need to be estimated are exceeded by the number of correlations, the model is defined. This is not the case, however, for *factor scores*. There will always be more free parameters to estimate than there will be observed data points. This leads to the problem of *factor indeterminacy*.

Given the data matrix, ${}_n\mathbf{X}_v$ with elements X_{ij}, we can find the deviation matrix, ${}_n\mathbf{x}_v$ and the covariance matrix between the observed variables, ${}_v\mathbf{C}_v$. This matrix may be reproduced by the product of the factor matrix and its transpose. See Equation 6.3 which is reproduced here with subscripts to show the number of observed covariances ($\frac{v(v-1)}{2}$) and the number of parameters to find ($vk - \frac{f*(f-1)}{2}$):

$$ {}_v\mathbf{C}_v \approx_v \mathbf{F}_{f\cdot f}\mathbf{F}'_v +_v \mathbf{U}^2_v \tag{6.24}$$

where the ${}_v\mathbf{U}^2_v$ values are found by subtracting the diagonal of the modeled covariances from the diagonal of the observed covariances.

We need to find *factor scores*, ${}_n\mathbf{S}_f$, and the uniquenesses, ${}_n\mathbf{U}_v$ such that

$$ {}_n\mathbf{x}_v =_n \mathbf{S}_{f\cdot f}\mathbf{F}'_v +_n \mathbf{U}_v = ({}_v\mathbf{F}_{f\cdot f}\mathbf{S}'_n +_v \mathbf{U}_n)' \tag{6.25}$$

and

$${}_v\mathbf{x}'_{n\cdot n}\mathbf{x}_v/n = ({}_v\mathbf{F}_{f\cdot f}\mathbf{S}'_n)({}_n\mathbf{S}_{f\cdot f}\mathbf{F}'_v)/n =_v \mathbf{C}_v.$$

Dropping the subscripts to make the equation more legible we find

$$\mathbf{C} = \mathbf{x}'\mathbf{x}/n = \mathbf{F}\mathbf{S}'\mathbf{S}\mathbf{F}'$$

and because the factors are orthogonal, $\mathbf{S}'\mathbf{S} = \mathbf{I}$,

$$\mathbf{C} = \mathbf{x}'\mathbf{x}/n = \mathbf{F}\mathbf{F}'.$$

Unfortunately, the number of free parameters in Equation 6.25 is $nv + nf + fv$ which, of course, exceeds the number of observed data points nv. That is, we have more unknowns than knowns and there are an infinity of possible solutions. We can see the problem more directly when solving solving equation 6.25 for $\hat{\mathbf{S}}$. Post-multiplying each side of the equation by ${}_v\mathbf{F}_f$ leads to

$$({}_n\mathbf{x}_v -_n \mathbf{U}_v)_v\mathbf{F}_f = {}_n\hat{\mathbf{S}}_{f\cdot f}\mathbf{F}'_{v\cdot v}\mathbf{F}_f = {}_n\hat{\mathbf{S}}_{f\cdot f}\mathbf{C}_f$$

and thus

$$\hat{\mathbf{S}} = (\mathbf{x} - \mathbf{U})\mathbf{F}\mathbf{C}^{-1} = \mathbf{x}\mathbf{F}\mathbf{C}^{-1} - \mathbf{U}\mathbf{F}\mathbf{C}^{-1}. \tag{6.26}$$

The problem of finding *factor score estimates*, $\hat{\mathbf{S}}$, is that while there is a observable part, $\mathbf{x}\mathbf{F}\mathbf{C}^{-1}$, of the score $\mathbf{S}$, there is also an unobservable part, $\mathbf{U}\mathbf{F}\mathbf{C}^{-1}$. Unless the communalities are one and therefore the uniquenesses are zero (that is, unless we are doing a *components*

analysis, the factor scores are indeterminant, although a best *estimate* of the factor scores (in terms of a least squares regression) will be

$$\hat{\mathbf{S}} = \mathbf{xFC}^{-1} = \mathbf{xW}$$

where

$$\mathbf{W} = \mathbf{FC}^{-1} \tag{6.27}$$

is just a matrix of the β weights for estimating factor scores from the observed variables and R^2 between these estimates and factors is

$$R^2 = diag(\mathbf{WF}). \tag{6.28}$$

Unfortunately, even for uncorrelated factors, the regression based weights found by Equation 6.27 will not necessarily produce uncorrelated factor score estimates. Nor, if the factors are correlated, will the factor scores have the same correlations. Gorsuch (1983) and Grice (2001) review several alternative ways to estimate factor scores, some of which will preserve orthogonality if the factors are orthogonal, others that will preserve the correlations between factors.

The regression solution (Equation 6.27) will produce factor score estimates that are most correlated with the factors, but will not preserve the factor intercorrelations (or, in the case of orthogonal factors, their orthogonality).

Harman (1976) proposed to weight the factor loadings based upon *idealized variables* by finding the inverse of the inner product of the factor loadings:

$${}_{\mathbf{k}}\mathbf{W}_{\mathbf{v}} = ({}_{\mathbf{k}}\mathbf{F}_{\mathbf{v}}{}'{}_{\mathbf{v}}\mathbf{F}_{\mathbf{k}})^{-1}{}_{\mathbf{k}}\mathbf{F}_{\mathbf{v}}. \tag{6.29}$$

A somewhat different least squares procedure was proposed by Bartlett (1937) to minimize the contribution of the unique factors:

$$\mathbf{W} = \mathbf{U}^{-2}\mathbf{F}(\mathbf{F}'\mathbf{U}^{-2}\mathbf{F})^{-1}. \tag{6.30}$$

A variation of Equation 6.30 proposed by Anderson and Rubin (1956) requires that the factors be orthogonal, and preserves this orthogonality:

$$\mathbf{W} = \mathbf{U}^{-2}\mathbf{F}(\mathbf{F}'\mathbf{U}^{-2}\mathbf{C}\mathbf{U}^{-2}\mathbf{F})^{-1/2}. \tag{6.31}$$

This solution was generalized to the oblique factors by McDonald (1981) and then extended by ten Berge et al. (1999): For a structure matrix, $\mathbf{F}$, with intercorrelations of the factors, $\boldsymbol{\Phi}$, then let $\mathbf{L} = \mathbf{F}\boldsymbol{\Phi}^{1/2}$, and $\mathbf{D} = \mathbf{R}^{1/2}\mathbf{L}(\mathbf{L}'\mathbf{C}^{-1}\mathbf{L})^{-1/2}$, then

$$W = \mathbf{C}^{-1/2}\mathbf{D}\boldsymbol{\Phi}^{1/2}. \tag{6.32}$$

Two of these scoring procedures are available in the `factanal` function (*regression scores* and *Bartlett's weighted least squares scores*. All five may be found using the `fa` or `factor.scores` functions in *psych*.

So why are there so many ways to estimate factor scores? It is because factor scores are underidentified and all of these procedures are approximations. Factor scores represent *estimates* of the common part of the variables and are not identical to the factors themselves. If a factor score estimate is thought of as a chop stick stuck into the center of an ice cream

cone and possible factor scores are represented by straws anywhere along the edge of the cone the problem of *factor indeterminacy* becomes clear, for depending on the shape of the cone, two straws can be negatively correlated with each other (Figure 6.14).

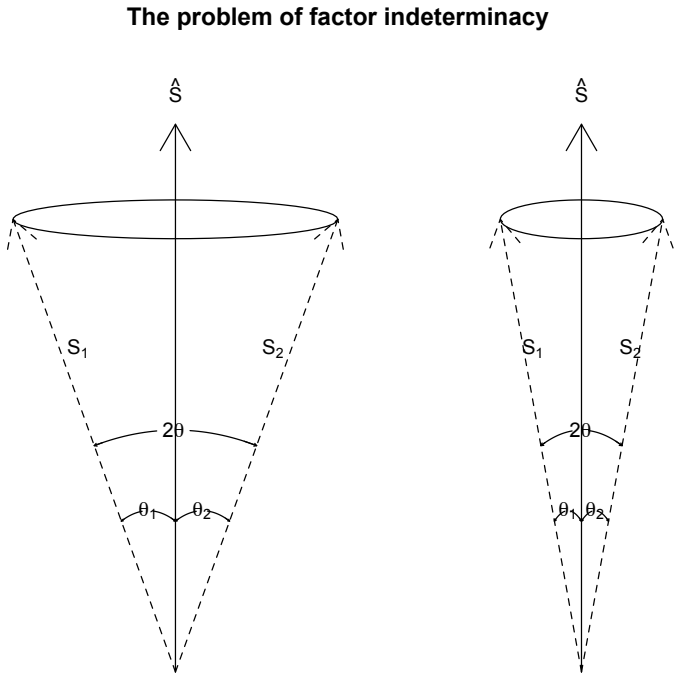

Fig. 6.14 Factor indeterminacy may be thought of as a chop stick ($\hat{S}$, the factor score estimates) inserted into an ice cream cone (the infinite set of factor scores that could generate the factor score estimates). In both figures, the vertical arrow is the factor score estimate, $\hat{S}$. The dashed arrows represent two alternative true factor score vectors, $\mathbf{S}_1$ and $\mathbf{S}_2$ that both correlate $cos(\theta_i)$ with the factor score estimate but $cos(2\theta) = 2cos(\theta)^2 - 1$ with each other. The imagery is taken from Niels Waller, adapted from Stanley Mulaik. The figure is drawn using the `dia.cone` function.

In a very clear discussion of the problem of *factor score indeterminacy*, Grice (2001) reviews these alternative ways of estimating factor scores and considers weighting schemes that will produce uncorrelated factor score estimates as well as the effect of using *course coded* (*unit weighted*) factor weights. From the trigonometric identity that $cos(2\theta) = cos(\theta)^2 - 1$ it follows that the minimum correlation of any two factor score vectors with each other is $2R^2 - 1$. That is, if the factor score estimates correlates .707 with the factor, then two sets of actual factor scores associated with these estimates could be uncorrelated! It is important to examine this level of indeterminacy for an $R^2 < .5$ implies that two estimates actually can be negatively correlated. The R^2 and minimum correlation possible between two factor scores

are reported in the output of the `fa`, `factor.scores` and `factor.stats` functions and may be seen in Tables 6.7 and 6.12.

It is appropriate to ask what affects the correlation of the factors with their factor scores? Just as with any regression, the larger the beta weights and the greater the number of variables, the better the fit. Examining Equation 6.26 it is clear that the score estimates will better match the factor scores as the contribution of the unique part of each variable diminishes. That is, higher communalities will decrease the indeterminacy. The limit of this is just a components model, in which component scores are perfectly defined by the data.

A demonstration that factor scores estimates are correlated with but not identical to the true factor scores can be shown by simulation. Consider the four variables examined in Table 6.16. The `sim.congeneric` function that generated the observed scores did so by first generating the latent variable and then adding error. Thus, it is possible to correlate the estimated scores with the "true" (but simulated) factor score (Table 6.23). The correlation, although large (.88) is not unity. It is the square root of the reliability of the factor score estimate.

Clusters and principal components, on the other hand, are defined. They are just weighted linear sums of the variables and thus are found by addition. Both are defined by

$$\mathbf{S} = \mathbf{xW}$$

where the weights matrix, $\mathbf{W}$, is formed of -1, 0, or 1s in the case of clusters, and are β weights found from the loadings and the inverse of the correlation matrix (Equation 6.27) in the case of principal components. In the case of the four correlated variables in Table 6.23, these two ways of finding scores are very highly correlated (.9981) as they are with the factor score estimates (.9811 and .9690 for components and clusters, respectively).

The differences between the various estimation procedures are shown clearly in Table 6.24 where 1000 subjects were simulated with a three correlated factor structure. Although the three factor solutions are identical (not shown), the factor scores estimates based upon the regression, Bartlett, or ten Berge methods are very different. Also shown are scores based upon a principal components solution and the simple expedient of *unit weighting* the items (that is, just adding them up).

That factor scores are indeterminate has been taken by some (e.g., Schonemann, 1996) to represent psychopathology on the part of psychometricians, for the problem of indeterminacy has been known (and either ignored or suppressed) since Wilson (1928). To others, factor indeterminacy is a problem at the data level but not the structural level, and at the data level it is adequate to report the degree of indeterminacy. This degree of indeterminacy is indeed striking (Schonemann and Wang, 1972; Velicer and Jackson, 1990b), and should be reported routinely. It is reported for each factor in the `fa` and `omega` functions.

6.9.1 Extending a factor solution to new variables

It is sometimes the case that factors are derived from a set of variables (the $vecF_o$ factor loadings) and we want to see what the loadings of an extended set of variables $\mathbf{F}_e$ would be. Given the original correlation matrix $\mathbf{R}_o$ and the correlation of these original variables with the extension variables of $\mathbf{R}_{oe}$, it is a straight forward calculation to find the loadings $\mathbf{F}_e$ of the extended variables on the original factors. This technique was developed by Dwyer (1937)

Table 6.23 Factor scores are estimated by equation 6.26 and are correlated with the "true" factors. Although the true scores are not normally available, by doing simulations it is possible to observe the correlation between true and estimated scores. The first variable is the latent factor score, e1 ... e4 are the error scores that went into each observed score, V1 ... V4 are the observed scores, MR1 is the factor score estimate, PC1 is the principal component score, A1 is the cluster score. Note the very high correlation between the factor estimates and principal component and cluster score, and the somewhat lower, but still quite large correlations of all three with the true, latent variable, theta.

```
> set.seed(42)
> test <- sim.congeneric(N=500,short=FALSE,loads = c(.8,.7,.6,.5))
> f1=fa(test$observed,1,scores=TRUE)
> c.scores <- score.items(rep(1,4),test$observed)
> pc1 <- principal(test$observed,1,scores=TRUE)
> scores.df <- data.frame(test$latent,test$observed,f1$scores,pc1$scores,c.scores$scores)
> round(cor(scores.df),2)
```

	theta	e1	e2	e3	e4	V1	V2	V3	V4	MR1	PC1	A1
theta	1.00	-0.01	0.04	0.00	-0.04	0.78	0.72	0.59	0.47	0.88	0.88	0.87
e1	-0.01	1.00	0.00	-0.02	-0.01	0.62	-0.01	-0.02	-0.02	0.35	0.22	0.20
e2	0.04	0.00	1.00	-0.04	-0.08	0.03	0.73	-0.01	-0.05	0.25	0.26	0.24
e3	0.00	-0.02	-0.04	1.00	0.00	-0.01	-0.03	0.81	0.00	0.16	0.25	0.26
e4	-0.04	-0.01	-0.08	0.00	1.00	-0.04	-0.08	-0.02	0.87	0.06	0.19	0.24
V1	0.78	0.62	0.03	-0.01	-0.04	1.00	0.56	0.45	0.35	0.91	0.83	0.81
V2	0.72	-0.01	0.73	-0.03	-0.08	0.56	1.00	0.40	0.28	0.79	0.78	0.76
V3	0.59	-0.02	-0.01	0.81	-0.02	0.45	0.40	1.00	0.27	0.65	0.72	0.72
V4	0.47	-0.02	-0.05	0.00	0.87	0.35	0.28	0.27	1.00	0.50	0.60	0.65
MR1	0.88	0.35	0.25	0.16	0.06	0.91	0.79	0.65	0.50	1.00	0.98	0.97
PC1	0.88	0.22	0.26	0.25	0.19	0.83	0.78	0.72	0.60	0.98	1.00	1.00
A1	0.87	0.20	0.24	0.26	0.24	0.81	0.76	0.72	0.65	0.97	1.00	1.00

for the case of adding new variables to a factor analysis without doing all the work over again and extended by Mosier (1938) to the multiple factor case. But, factor extension is also appropriate when one does not want to include the extension variables in the original factor analysis, but does want to see what the loadings would be anyway (Horn, 1973; Gorsuch, 1997). Logically, this is correlating factor scores with these new variables, but in fact can be done without finding the factor scores.

Consider the super matrix $\mathbf{R}$ made up of the correlations of the variables within the original set, $\mathbf{R}_o$, the correlations between the variables in the two sets, $\mathbf{R}_{oe}$, and the correlations of the variables within the second set, $\mathbf{R}_o$ (the left hand part of Equation 6.33. This can be factored the normal way (the right hand part of the equation with θ being the factor intercorrelatons).

$$\mathbf{R} = \begin{pmatrix} \mathbf{R}_o & \vdots & \mathbf{R}'_{oe} \\ \cdots & \cdots & \cdots \\ \mathbf{R}_{oe} & \vdots & \mathbf{R}_e \end{pmatrix} = \begin{pmatrix} \mathbf{F}_o \\ \cdots \\ \mathbf{F}_e \end{pmatrix} \theta \begin{pmatrix} \mathbf{F}'_o & \vdots & \mathbf{F}'_e \end{pmatrix} + \mathbf{U}^2 \tag{6.33}$$

Now $\mathbf{R}_o = \mathbf{F}_o \theta \mathbf{F}'_o$ and $\mathbf{R}_{oe} = \mathbf{F}_o \theta \mathbf{F}'_e$. Thus,

$$\mathbf{F}_e = \mathbf{R}_{oe} \mathbf{F}_o (\mathbf{F}'_o \mathbf{F}_o)^{-1} \theta^{-1}. \tag{6.34}$$

Table 6.24 9 variables were generated with a correlated factor structure. Three methods of factor score estimation produce very different results. The regression scoring method does not produce factor score estimates with the same correlations as the factors and does not agree with the estimates from either Bartlett or tenBerge. The tenBerge estimates have the identical correlations as the factors. The last two methods are component scores (RC1 ... RC3) and unit weighted "cluster" scores.

```
> set.seed(42)
> v9 <- sim.hierarchical(n=1000,raw=TRUE)$observed
> f3r <- fa(v9,3,scores="regression") #the default
> f3b <- fa(v9,3,scores="Bartlett")
> f3t <- fa(v9,3,scores="tenBerge")
> p3 <- principal(v9,3,scores=TRUE,rotate="oblimin")
> keys <- make.keys(9,list(C1=1:3,C2=4:6,C3=5:7))
> cluster.scores <- score.items(keys,v9)
> scores <- data.frame(f3r$scores,f3b$scores,f3t$scores,p3$scores,cluster.scores$scores)
> round(cor(scores),2)
```

	MR1	MR2	MR3	MR1.1	MR2.1	MR3.1	MR1.2	MR2.2	MR3.2	RC1	RC3	RC2	C1	C2	C3
MR1	1.00	-0.39	-0.23	0.87	0.00	0.00	0.81	0.14	0.14	0.94	-0.16	-0.09	0.85	0.01	0.00
MR2	-0.39	1.00	-0.14	0.00	0.89	0.00	0.09	0.80	0.08	-0.21	0.93	-0.08	0.02	0.86	0.58
MR3	-0.23	-0.14	1.00	0.00	0.00	0.94	0.07	0.06	0.85	-0.12	-0.07	0.92	0.04	0.04	0.45
MR1.1	0.87	0.00	0.00	1.00	0.44	0.32	0.99	0.58	0.48	0.93	0.25	0.17	0.99	0.45	0.43
MR2.1	0.00	0.89	0.00	0.44	1.00	0.26	0.53	0.98	0.39	0.21	0.95	0.12	0.46	0.98	0.77
MR3.1	0.00	0.00	0.94	0.32	0.26	1.00	0.39	0.35	0.98	0.15	0.14	0.93	0.35	0.29	0.64
MR1.2	0.81	0.09	0.07	0.99	0.53	0.39	1.00	0.66	0.56	0.89	0.34	0.24	0.99	0.54	0.53
MR2.2	0.14	0.80	0.06	0.58	0.98	0.35	0.66	1.00	0.50	0.35	0.90	0.20	0.60	0.97	0.80
MR3.2	0.14	0.08	0.85	0.48	0.39	0.98	0.56	0.50	1.00	0.31	0.25	0.88	0.52	0.42	0.72
RC1	0.94	-0.21	-0.12	0.93	0.21	0.15	0.89	0.35	0.31	1.00	0.00	0.00	0.93	0.22	0.20
RC3	-0.16	0.93	-0.07	0.25	0.95	0.14	0.34	0.90	0.25	0.00	1.00	0.00	0.27	0.95	0.76
RC2	-0.09	-0.08	0.92	0.17	0.12	0.93	0.24	0.20	0.88	0.00	0.00	1.00	0.19	0.14	0.43
C1	0.85	0.02	0.04	0.99	0.46	0.35	0.99	0.60	0.52	0.93	0.27	0.19	1.00	0.47	0.47
C2	0.01	0.86	0.04	0.45	0.98	0.29	0.54	0.97	0.42	0.22	0.95	0.14	0.47	1.00	0.84
C3	0.00	0.58	0.45	0.43	0.77	0.64	0.53	0.80	0.72	0.20	0.76	0.43	0.47	0.84	1.00

Consider a case of twelve variables reflecting two correlated factors with the first six variables loading on the first factor, and the second six on the second factor. Let variables 3, 6, 9, and 12 form the extension variables. Factor the remaining variables and then extend these factors to the omitted variables using `fa.extension` (Table 6.25). This may be shown graphically using the `fa.diagram` function (Figure 6.15). The loadings are

6.9.2 Comparing factors and components – part 2

At the structural level, there are major theoretical differences between factors, components, and clusters. Factors are seen as the causes of the observed variables, clusters and components are just summaries of the observed variables. Factors represent the common part of variables, clusters and components all of the variables. Factors do not change with the addition of new variables, clusters and components do. Factor loadings will tend to be smaller than component loadings and in the limiting case of unrelated variables, can be zero, even though component loadings are large. At the scores level, factor scores are estimated from the data, while clusters and components are found.

Table 6.25 Create 12 variables with a clear two factor structure. Remove variables 3, 6, 9, and 12 from the matrix and factor the other variables. Then extend this solutio to the deleted variables. Compare this solution to the solution with all variables included (not shown). A graphic representation is in Figure 6.15.

```
set.seed(42)
fx <- matrix(c(.9,.8,.7,.85,.75,.65,rep(0,12),.9,.8,.7,.85,.75,.65),ncol=2)
 Phi <- matrix(c(1,.6,.6,1),2)
 sim.data <- sim.structure(fx,Phi,n=1000,raw=TRUE)
 R <- cor(sim.data$observed)
 Ro <- R[c(1,2,4,5,7,8,10,11),c(1,2,4,5,7,8,10,11)]
 Roe <- R[c(1,2,4,5,7,8,10,11),c(3,6,9,12)]
 fo <- fa(Ro,2)
 fe <- fa.extension(Roe,fo)
 fa.diagram(fo,fe=fe)
```

```
Call: fa.extension(Roe = Roe, fo = fo)
Standardized loadings based upon correlation matrix
      MR1   MR2   h2   u2
V3   0.69  0.01 0.49 0.51
V6   0.66 -0.02 0.42 0.58
V9   0.01  0.66 0.44 0.56
V12 -0.06  0.70 0.44 0.56

                 MR1  MR2
SS loadings    0.89 0.89
Proportion Var 0.22 0.22
Cumulative Var 0.22 0.45

 With factor correlations of
     MR1  MR2
MR1 1.00 0.62
MR2 0.62 1.00
```

Unfortunately, many of the theoretical distinctions between factor analysis and components analysis are lost when one of the major commercial packages claims to be doing factor analysis by doing principal components.

Basic concepts of reliability and structural equation modeling are built upon the logic of factor analysis and most discussions of theoretical constructs are discussions of factors rather than components. There is a reason for this. The pragmatic similarity between the three sets of models should not be used as an excuse for theoretical sloppiness.

In the next three chapters, the use of latent variable models using factor analysis and related concepts will be applied to the problem of how well is a trait measured (issues in reliability), alternative models for determining latent variables (item response theory), and the use of latent variable models to estimate the structural properties of the data (structural equation modeling). All of the techniques rely on a basic understanding of latent variable modeling and differ only in how the latent variables are estimated.

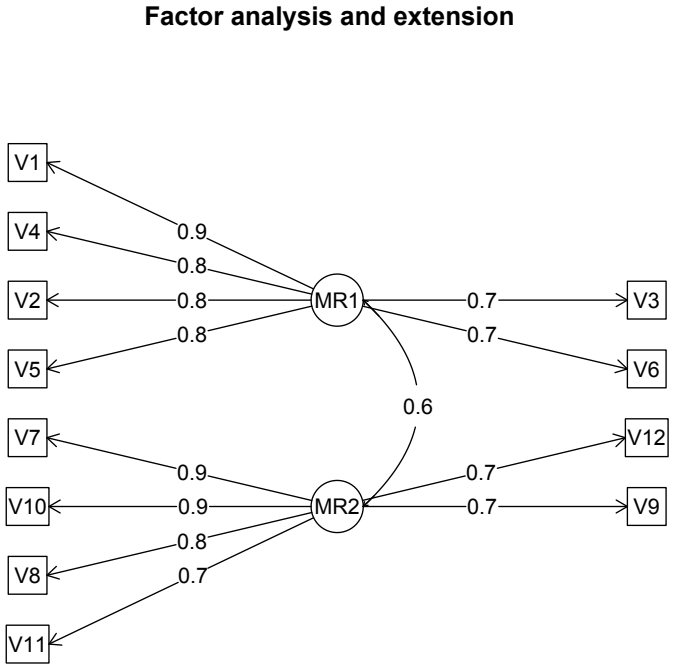

Fig. 6.15 Factor extension projects new variables into a factor solution from an original set of variables. In this simulated data set (see Table 6.25, 12 variables were generated to reflect two correlated factors. Variables 3, 6, 9, and 12 were removed and the remaining variables were factored using `fa`. The loadings of the removed variables on these original factors were then calculated using `fa.extension`. Compare these extended loadings to the loadings had all the variables been factored toegether.