Factor Analysis

Rachael Smyth and Andrew Johnson

Introduction

For this lab, we are going to explore the factor analysis technique, looking at both principal axis and principal components extraction methods, two different methods of identifying the correct number of factors to extract (scree plot and parallel analysis), and two different methods of rotating factors to facilitate interpretation.

Load Libraries

The libraries we'll need for this lab are `psych` and `GPArotation`.

```
library(psych)
library(GPArotation)
```

The Data

The dataset that we'll use for this demonstration is called `bfi` and comes from the `psych` package. It is made up of 25 self-report personality items from the International Personality Item Pool, gender, education level and age for 2800 subjects and used in the Synthetic Aperture Personality Assessment.

The personality items are split into 5 categories: Agreeableness (A), Conscientiousness (C), Extraversion (E), Neuroticism (N), Openness (O). Each item was answered on a six point scale: 1 Very Inaccurate, 2 Moderately Inaccurate, 3 Slightly Inaccurate, 4 Slightly Accurate, 5 Moderately Accurate, 6 Very Accurate.

```
data("bfi")
```

Describing the data

It's a good idea to look at your data before you run any analysis. Any participant who is missing any piece of data will be fully excluded from the analysis. It's important to keep that in mind.

```
describe(bfi[1:25])
```

```
##    vars    n mean   sd median trimmed  mad min max range  skew kurtosis   se
## A1    1 2784 2.41 1.41      2    2.23 1.48   1   6     5  0.83    -0.31 0.03
## A2    2 2773 4.80 1.17      5    4.98 1.48   1   6     5 -1.12     1.05 0.02
## A3    3 2774 4.60 1.30      5    4.79 1.48   1   6     5 -1.00     0.44 0.02
## A4    4 2781 4.70 1.48      5    4.93 1.48   1   6     5 -1.03     0.04 0.03
## A5    5 2784 4.56 1.26      5    4.71 1.48   1   6     5 -0.85     0.16 0.02
## C1    6 2779 4.50 1.24      5    4.64 1.48   1   6     5 -0.85     0.30 0.02
## C2    7 2776 4.37 1.32      5    4.50 1.48   1   6     5 -0.74    -0.14 0.03
## C3    8 2780 4.30 1.29      5    4.42 1.48   1   6     5 -0.69    -0.13 0.02
## C4    9 2774 2.55 1.38      2    2.41 1.48   1   6     5  0.60    -0.62 0.03
## C5   10 2784 3.30 1.63      3    3.25 1.48   1   6     5  0.07    -1.22 0.03
```

```
## E1   11 2777 2.97 1.63      3    2.86 1.48   1   6     5  0.37    -1.09 0.03
## E2   12 2784 3.14 1.61      3    3.06 1.48   1   6     5  0.22    -1.15 0.03
## E3   13 2775 4.00 1.35      4    4.07 1.48   1   6     5 -0.47    -0.47 0.03
## E4   14 2791 4.42 1.46      5    4.59 1.48   1   6     5 -0.82    -0.30 0.03
## E5   15 2779 4.42 1.33      5    4.56 1.48   1   6     5 -0.78    -0.09 0.03
## N1   16 2778 2.93 1.57      3    2.82 1.48   1   6     5  0.37    -1.01 0.03
## N2   17 2779 3.51 1.53      4    3.51 1.48   1   6     5 -0.08    -1.05 0.03
## N3   18 2789 3.22 1.60      3    3.16 1.48   1   6     5  0.15    -1.18 0.03
## N4   19 2764 3.19 1.57      3    3.12 1.48   1   6     5  0.20    -1.09 0.03
## N5   20 2771 2.97 1.62      3    2.85 1.48   1   6     5  0.37    -1.06 0.03
## O1   21 2778 4.82 1.13      5    4.96 1.48   1   6     5 -0.90     0.43 0.02
## O2   22 2800 2.71 1.57      2    2.56 1.48   1   6     5  0.59    -0.81 0.03
## O3   23 2772 4.44 1.22      5    4.56 1.48   1   6     5 -0.77     0.30 0.02
## O4   24 2786 4.89 1.22      5    5.10 1.48   1   6     5 -1.22     1.08 0.02
## O5   25 2780 2.49 1.33      2    2.34 1.48   1   6     5  0.74    -0.24 0.03
```

This suggests that most items are only missing 20 or 30 participants worth of data - which is no big deal in a data set with 2800 observations. It is possible, however, that some of these missing values are non-overlapping - meaning that it could be a different 20 or 30 individuals missing from each of the variables. We can, however, determine the number of "complete cases" within the data - these are individuals that are missing no data whatsoever on the questionnaire.

```
sum(complete.cases(bfi[1:25]))
```

```
## [1] 2436
```

The `complete.cases` function generates a Boolean vector, where a value of "TRUE" means that the case is complete, and a value of "FALSE" means that the case is missing at least one value. Summing across this vector gives us the total number of complete cases. This means that there are 2436 cases with no missing data. This means that 13% of the data is missing. There is no magic number as to the amount of missing data that is acceptable - but sample size is important for factor analysis. Some authors suggest that you need at least 10 observations for each variable in the factor analysis - our sample size is, therefore, adequate for our purposes.

Assessing the Factorability of the Data

Before we go too far down the road with this analysis, we should evaluate the "factorability" of our data. In other words, "are there meaningful latent factors to be found within the data?" We can check two things: (1) Bartlett's test of sphericity; and (2) the Kaiser-Meyer-Olkin measure of sampling adequacy.

Bartlett's Test of Sphericity

The most liberal test is Bartlett's test of sphericity - this evaluates whether or not the variables intercorrelate at all, by evaluating the observed correlation matrix against an "identity matrix" (a matrix with ones along the principal diagonal, and zeroes everywhere else). If this test is not statistically significant, you should not employ a factor analysis.

```
cortest.bartlett(bfi[1:25])
```

```
## R was not square, finding R from data
```

```
## $chisq
## [1] 20163.79
##
## $p.value
## [1] 0
##
## $df
## [1] 300
```

Bartlett's test was statistically significant, suggesting that the observed correlation matrix among the items is not an identity matrix. This really isn't a particularly powerful indication that you have a factorable dataset, though - all it really tells you is that at least *some* of the variables are correlated with each other.

KMO

The Kaiser-Meyer-Olkin (KMO) measure of sampling adequacy is a better measure of factorability. The KMO tests to see if the partial correlations within your data are close enough to zero to suggest that there is at least one latent factor underlying your variables. The minimum acceptable value is 0.50, but most authors recommend a value of at 0.60 before undertaking a factor analysis. The `KMO` function in the `psych` package produces an overall Measure of Sampling Adequacy (MSA, as its labelled in the output), and an MSA for each item. Theoretically, if your overall MSA is too low, you could look at the item MSA's and drop items that are too low. This should be done with caution, of course, as is the case with any atheoretical, empirical method of item selection.

```
KMO(bfi[1:25])
```

```
## Kaiser-Meyer-Olkin factor adequacy
## Call: KMO(r = bfi[1:25])
## Overall MSA =  0.85
## MSA for each item =
##   A1   A2   A3   A4   A5   C1   C2   C3   C4   C5   E1   E2   E3   E4   E5   N1   N2   N3
## 0.74 0.84 0.87 0.87 0.90 0.83 0.79 0.85 0.82 0.86 0.83 0.88 0.89 0.87 0.89 0.78 0.78 0.86
##   N4   N5   O1   O2   O3   O4   O5
## 0.88 0.86 0.85 0.78 0.84 0.76 0.76
```

The overall KMO for our data is 0.85 which is excellent - this suggests that we can go ahead with our planned factor analysis.

Determining the Number of Factors to Extract

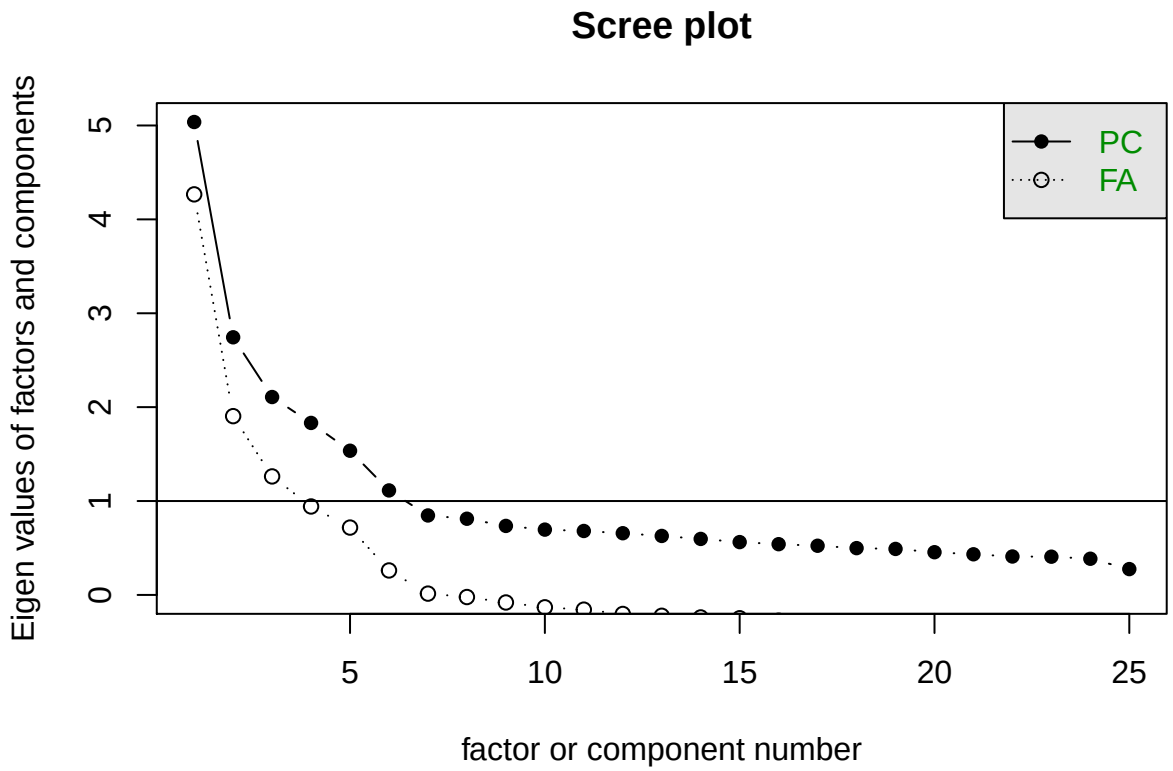
The first decision that we will face in our factor analysis is the decision as to the number of factors that we will need to extract, in order to achieve the most parsimonious (but still interpretatable) factor structure. There are a number of methods that we could use, but the two most commonly employed methods are the *scree plot*, and *parallel analysis*. The simplest technique is the scree plot.

Scree Plot

Eigenvalues are a measure of the amount of variance accounted for by a factor, and so they can be useful in determining the number of factors that we need to extract. In a scree plot, we simply plot the eigenvalues for all of our factors, and then look to see where they drop off sharply.

Let's take a look at the scree plot for the bfi dataset:

```
scree(bfi[,1:25])
```



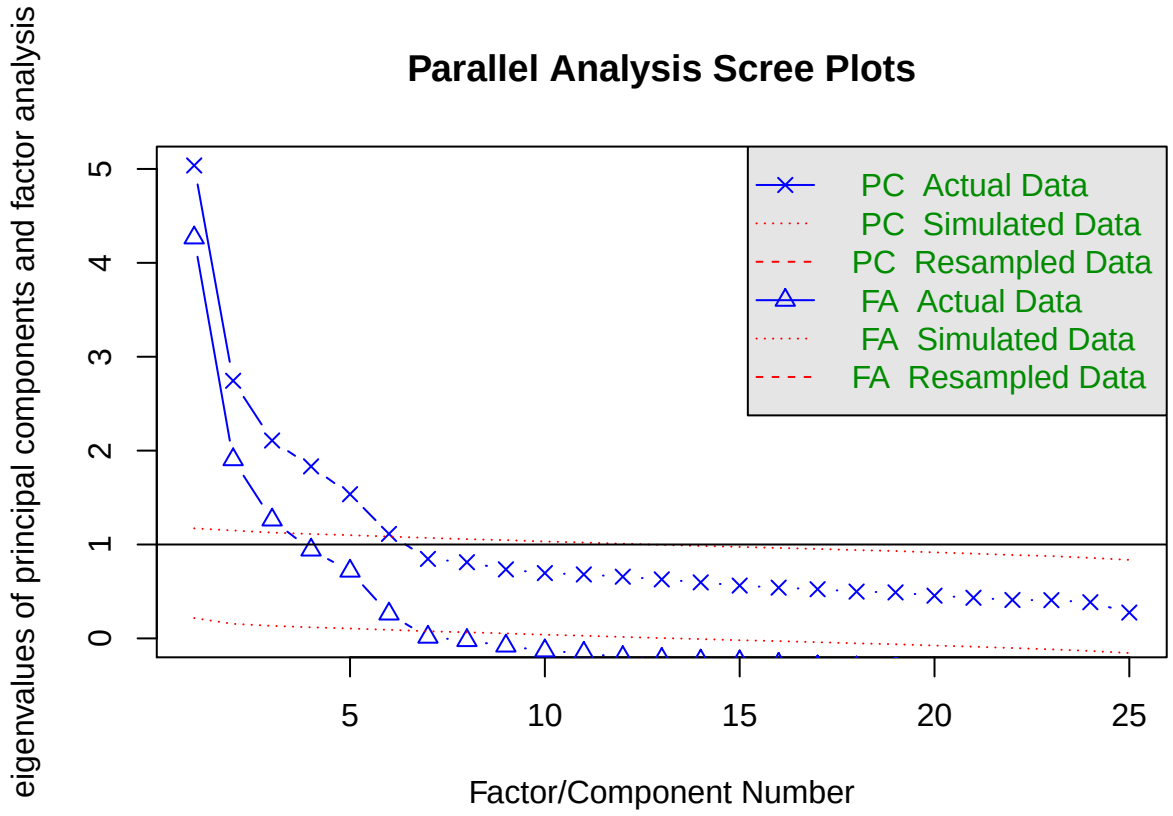
The scree plot technique involves drawing a straight line through the plotted eigenvalues, starting with the largest one. The last point to fall on this line represents the last factor that you extract, with the idea being that beyond this, the amount of additional variance explained is non-meaningful. In fact, the word "scree" refers to the loose stone that lies around the base of the mountain. A "scree plot" is effectively looking to help you differentiate between the points that represent "mountain", and the points that represent "scree."

Regardless of whether you are using a principal components or a principal axis factor extraction, however, there is a very large first factor in this data. If we were to draw our straight line starting at this point, you would probably conclude that there are only three factors in the dataset. If, however, you were to start your line at the second point in the scree plot, you would probably conclude that there are five factors in the dataset. The latter interpretation is probably closer to the truth, but if this were the only piece of evidence brought to bear in our consideration of the number of factors to extract, we might want to look at both of these factor solutions.

Parallel Analysis

A better method for evaluating the scree plot is within a parallel analysis. In addition to plotting the eigenvalues from our factor analysis (whether it's based on principal axis or principal components extraction), a parallel analysis involves generating random correlation matrices and after factor analyzing them, comparing the resulting eigenvalues to the eigenvalues of the observed data. The idea behind this method is that observed eigenvalues that are higher than their corresponding random eigenvalues are more likely to be from "meaningful factors" than observed eigenvalues that are below their corresponding random eigenvalue.

```
fa.parallel(bfi[1:25])
```




```
## Parallel analysis suggests that the number of factors =  6  and the number of components =  6
```

When looking at the parallel analysis scree plots, there are two places to look depending on which type of factor analysis you're looking to run. The two blue lines show you the observed eigenvalues - they should look identical to the scree plots drawn by the `scree` function. The red dotted lines show you the random eigenvalues or the simulated data line. Each point on the blue line that lies above the corresponding simulated data line is a factor or component to extract. In this analysis, you can see that 6 factors in the "Factor Analysis" parallel analysis lie above the corresponding simulated data line and 6 components in the "Principal Components" parallel analysis lie above the corresponding simulated data line.

In our case, however, the last factor/component lies very close to the line - for both principal components extraction and principal axis extraction. Thus, we should probably compare the 6 factor and 5 factor solutions, to see which one is most interpretable.

Conducting the Factor Analysis

We already have a good idea as to how many factors (5 or 6) that we should extract in our analysis of the `bfi` data object. Now we need to decide whether we will use "common factor" analysis, or "principal components" analysis. In a very broad sense, "common factor" analysis (or "principal axis factoring") is used when we want to identify the latent variables that are underlying a set of variables, while "principal components" analysis is used to reduce a set of variables to a smaller set of factors (i.e., the "principal components" of the data). In other words, common factor analysis is used when you want to evaluate a theoretical model with a set of variables, and principal components analysis is used for data reduction.

Both of these approaches have merit in test construction, and so we will walk through each approach with this data.

Principal Axis Factoring (Common Factor Analysis)

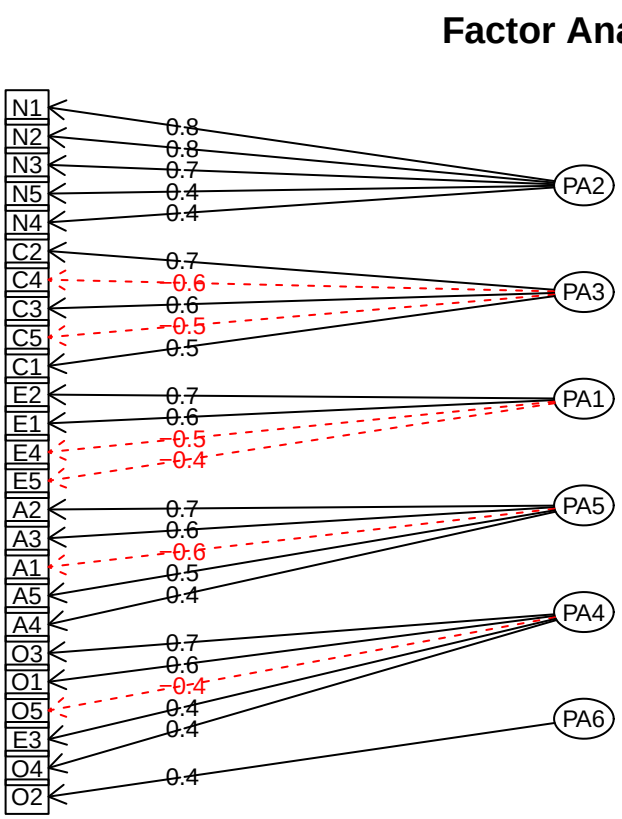
The `fa` function takes the following parameters when it is called:

- the variables to be used within the factor analysis (items 1-25 from the `bfi` object)
- the number of factors we want to extract (6)
- the type of factor analysis we want to use ("pa" is principal axis factoring)
- the number of iterations or attempts to use when identifying the "best"" solution (50 is the default, but we have changed it to 100)
- the type of rotation we want to use (we'll start with "oblimin")

```
pa6.out <- fa(bfi[1:25],
              nfactors = 6,
              fm="pa",
              max.iter = 100,
              rotate = "oblimin")
```

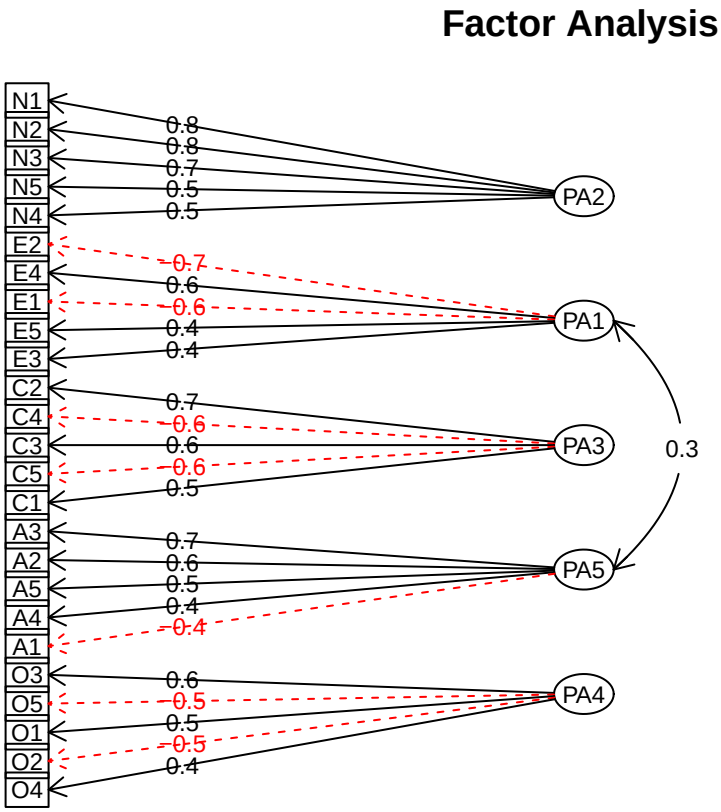
A quick way to visualize your rotated factor solution, and determine whether it represents an "interpretable" solution is to use the `fa.diagram` function.

```
fa.diagram(pa6.out)
```



As you can see from this, the sixth factor has only one variable loading on it - the second item on the openness to experience scale. Thus, this probably represents an overextraction... let's take a look at the five factor solution.

```
pa5.out <- fa(bfi[1:25],
              nfactors = 5,
              fm="pa",
              max.iter = 100,
              rotate = "oblimin")
fa.diagram(pa5.out)
```



The five-factor solution is more interpretable - in fact, it seems to replicate the expected factor structure nicely.

Communalities

The communality for each variable is the percentage of variance that can be explained by the retained factors. It's best if the retained factors explain more of the variance in each variable.

```
pa5.out$communality
```

```
##        A1        A2        A3        A4        A5        C1        C2        C3        C4
## 0.1917679 0.4472702 0.5226804 0.2800492 0.4638037 0.3301446 0.4502136 0.3182264 0.4506858
##        C5        E1        E2        E3        E4        E5        N1        N2        N3
## 0.4272571 0.3479662 0.5432814 0.4389546 0.5313933 0.4026775 0.6517914 0.6000981 0.5471437
##        N4        N5        O1        O2        O3        O4        O5
## 0.4881259 0.3496557 0.3126709 0.2575010 0.4639493 0.2512316 0.3002371
```

As a point of interest, the primary difference between the way that common factor analysis and principal component analysis are conducted, is that the correlation matrix on which the factor analysis is based has

ones along the principal diagonal in principal components analysis, and the communalities along the principal diagonal in principal axis factor analysis.

Eigenvalues

The eigenvalues derived in the extracted factor solution are stored within `e.values`. These are the eigenvalues that were plotted in the scree plots that we looked at near the beginning of this process.

```
pa5.out$e.values[1:5]
```

```
## [1] 5.036903 2.744085 2.107632 1.831842 1.535686
```

If you want the eigenvalues from the rotated solution, you would ask for `values`.

```
pa5.out$values[1:5]
```

```
## [1] 4.4928257 2.2485818 1.5051927 1.1878333 0.9343431
```

Percentage of Variance Accounted For

We can use the eigenvalues to calculate the percentage of variance accounted for by each of the factors. Given that the maximum sum of the eigenvalues will always be equal to the total number of variables in the analysis, we can calculate the percentage of variance accounted for by dividing each eigenvalue by the total number of variables in the analysis. In our example this is 25.

```
100*pa5.out$e.values[1:5]/length(pa5.out$e.values)
```

```
## [1] 20.147610 10.976342  8.430529  7.327366  6.142746
```

If you wanted the percentage of variance accounted for by the rotated solution, you would use the eigenvalues stored in `values` rather than `e.values`.

```
100*pa5.out$values[1:5]/length(pa5.out$values)
```

```
## [1] 17.971303  8.994327  6.020771  4.751333  3.737372
```

Rotated Solution

We've already peeked at the highest-loading items for each factor (using `fa.diagram`), but this only tells us the largest loading for each item. Each item will, however, load on each of the factors to a greater or lesser degree - and we will eventually want to look at the full factor loading matrix. The factor loading matrix shows us the factor loadings for each variable, after they have been rotated to "simple structure." Essentially, we are taking advantage of the fact that there are a number of factor solutions that are equally acceptable to the "optimal" solution that was found within our initial extraction (i.e., that are mathematically equivalent), and rotating the factors so that they are more easily interpreted.

Because we have used an oblique factor rotation ("oblimin"), this is sometimes (e.g., in SPSS) called a pattern matrix.

```
print(pa5.out$loadings, cutoff=0, digits=3)
```

```
##
## Loadings:
##    PA2    PA1    PA3    PA5    PA4
## A1  0.213  0.166  0.067 -0.414 -0.058
## A2 -0.023 -0.002  0.077  0.640  0.032
## A3 -0.029  0.116  0.025  0.660  0.031
## A4 -0.057  0.065  0.193  0.433 -0.148
## A5 -0.112  0.234  0.006  0.532  0.044
## C1  0.069 -0.027  0.546 -0.023  0.148
## C2  0.149 -0.085  0.666  0.081  0.039
## C3  0.034 -0.061  0.567  0.092 -0.068
## C4  0.174  0.002 -0.614  0.040 -0.048
## C5  0.189 -0.142 -0.553  0.018  0.092
## E1 -0.059 -0.557  0.106 -0.083 -0.102
## E2  0.099 -0.676 -0.016 -0.048 -0.058
## E3  0.083  0.418 -0.001  0.245  0.283
## E4  0.013  0.591  0.024  0.287 -0.077
## E5  0.152  0.421  0.272  0.052  0.206
## N1  0.814  0.103  0.004 -0.111 -0.047
## N2  0.777  0.040  0.011 -0.094  0.015
## N3  0.707 -0.100 -0.035  0.079  0.023
## N4  0.474 -0.386 -0.135  0.095  0.080
## N5  0.486 -0.202 -0.004  0.207 -0.150
## O1  0.018  0.103  0.073  0.015  0.508
## O2  0.195  0.057 -0.078  0.163 -0.456
## O3  0.031  0.152  0.017  0.083  0.609
## O4  0.126 -0.323 -0.024  0.174  0.371
## O5  0.132  0.098 -0.025  0.043 -0.542
##
##                  PA2   PA1   PA3   PA5   PA4
## SS loadings    2.499 1.964 1.913 1.804 1.511
## Proportion Var 0.100 0.079 0.077 0.072 0.060
## Cumulative Var 0.100 0.179 0.255 0.327 0.388
```

We can also look at the structure matrix - this is just the pattern matrix multiplied by the factor intercorrelation matrix. The result is that these values represent the correlations between the variables and the factors - which may be more intuitive to interpret.

```
print(pa5.out$Structure, cutoff=0, digits=3)
```

```
##
## Loadings:
##    [,1]   [,2]   [,3]   [,4]   [,5]
## A1  0.181 -0.010 -0.030 -0.365 -0.100
## A2 -0.061  0.237  0.217  0.662  0.171
## A3 -0.083  0.350  0.197  0.710  0.183
## A4 -0.121  0.239  0.277  0.467 -0.016
## A5 -0.183  0.441  0.197  0.622  0.188
## C1 -0.028  0.100  0.551  0.105  0.244
## C2  0.039  0.070  0.643  0.191  0.168
## C3 -0.062  0.081  0.552  0.173  0.050
## C4  0.287 -0.171 -0.647 -0.100 -0.162
## C5  0.321 -0.288 -0.600 -0.131 -0.038
## E1  0.044 -0.564 -0.047 -0.262 -0.190
## E2  0.248 -0.726 -0.211 -0.289 -0.184
## E3 -0.018  0.528  0.185  0.434  0.399
## E4 -0.127  0.675  0.201  0.471  0.081
## E5  0.007  0.502  0.390  0.280  0.337
## N1  0.796 -0.114 -0.156 -0.116 -0.059
## N2  0.770 -0.152 -0.141 -0.105 -0.003
## N3  0.731 -0.229 -0.170  0.016  0.007
## N4  0.577 -0.474 -0.278 -0.062  0.003
## N5  0.524 -0.263 -0.128  0.092 -0.149
## O1 -0.023  0.205  0.195  0.161  0.542
## O2  0.196 -0.025 -0.157  0.070 -0.432
## O3 -0.014  0.278  0.182  0.253  0.653
## O4  0.189 -0.237 -0.015  0.129  0.345
## O5  0.120 -0.012 -0.124 -0.039 -0.524
##
##                  [,1]  [,2]  [,3]  [,4]  [,5]
## SS loadings    2.821 2.956 2.554 2.577 1.902
## Proportion Var 0.113 0.118 0.102 0.103 0.076
## Cumulative Var 0.113 0.231 0.333 0.436 0.512
```

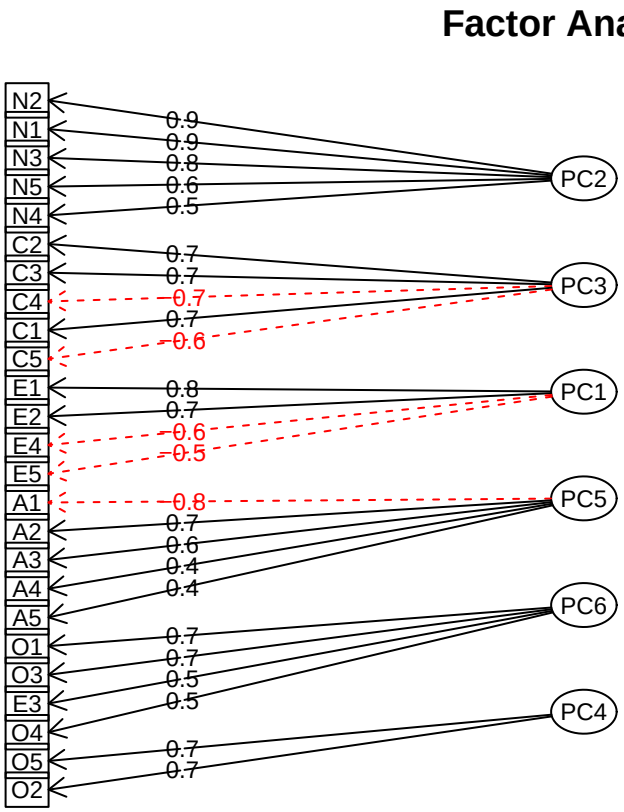
Principal Components Analysis

This is how you run a Principal Components Analysis in R. The command is not the same as running Principal Axis Factoring. Many of the steps will be the same, but we'll go through them for the Principal Components Analysis as well.

```
pc6.out <- principal(bfi[1:25],
                     nfactors = 6,
                     rotate = "oblimin")
```

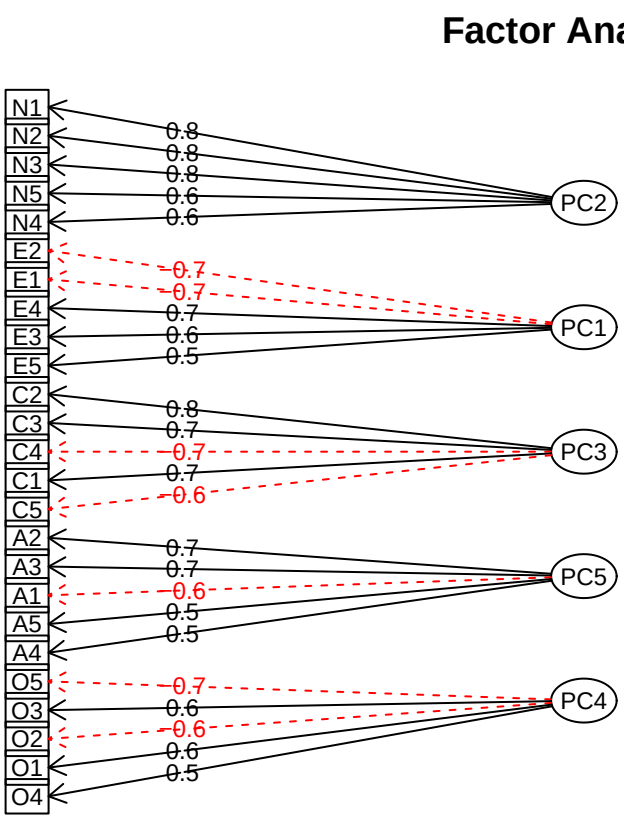
Again, we can take a quick look at the factor structure for this solution using `fa.diagram`.

```
fa.diagram(pc6.out)
```



As you can see, the six factor solution results in a factor that has major loadings from only two items, and is not easily interpreted. Let's take a look at the five factor solution.

```
pc5.out <- principal(bfi[1:25],
                     nfactors = 5,
                     rotate = "oblimin")
fa.diagram(pc5.out)
```



As was the case with the principal axis / common factor solution, the five-factor principal components solution is far more interpretable, aligning very nicely with the expected item-factor orientation.

Now that we have determined the number of factors to extract, we can look at all of the same information that we pulled out of the common factor solution.

Communalities

Again, these are the percentage of variance that can be explained by the retained factors for each variable.

```
pc5.out$communality
```

```
##        A1        A2        A3        A4        A5        C1        C2        C3        C4
## 0.4600865 0.5738357 0.5948168 0.4079652 0.5351829 0.4720674 0.5761450 0.4720012 0.5439312
##        C5        E1        E2        E3        E4        E5        N1        N2        N3
## 0.5235458 0.4755423 0.6048763 0.5303698 0.6061815 0.5018843 0.6886723 0.6594993 0.6342482
##        N4        N5        O1        O2        O3        O4        O5
## 0.5719617 0.4803649 0.4404534 0.4285102 0.5534135 0.4408476 0.4797453
```

Eigenvalues and % of Variance Accounted For

This computes a vector of the eigenvalues for our five principal components:

```
pc5.out$values[1:5]
```

```
## [1] 5.036903 2.744085 2.107632 1.831842 1.535686
```

and this uses those eigenvalues to compute the percentage of variance associated with each of these factors:

```
100*pc5.out$values[1:5]/length(pc5.out$values)
```

```
## [1] 20.147610 10.976342  8.430529  7.327366  6.142746
```

Rotated Solution

This factor loading matrix shows us the variables that load on each of the factors we have extracted...

```
print(pc5.out$loadings, cutoff=0, digits=3)
```

```
##
## Loadings:
##    PC2    PC1    PC3    PC5    PC4
## A1  0.228  0.232  0.109 -0.638 -0.098
## A2  0.023  0.093  0.093  0.710  0.023
## A3  0.011  0.249  0.041  0.669  0.010
## A4 -0.048  0.102  0.234  0.511 -0.209
## A5 -0.095  0.361  0.016  0.538  0.013
## C1  0.075 -0.012  0.660 -0.055  0.167
## C2  0.161 -0.058  0.755  0.063  0.037
## C3  0.034 -0.080  0.691  0.090 -0.089
## C4  0.239  0.042 -0.668  0.017 -0.070
## C5  0.267 -0.114 -0.611  0.024  0.106
## E1 -0.030 -0.690  0.150 -0.037 -0.068
## E2  0.178 -0.726 -0.004 -0.017 -0.016
## E3  0.097  0.586  0.004  0.201  0.259
## E4 -0.022  0.686  0.029  0.235 -0.133
## E5  0.137  0.541  0.303 -0.007  0.187
## N1  0.818  0.110 -0.002 -0.150 -0.059
## N2  0.805  0.061  0.009 -0.129  0.007
## N3  0.788 -0.041 -0.023  0.041  0.018
## N4  0.590 -0.355 -0.124  0.098  0.101
## N5  0.609 -0.181  0.024  0.223 -0.178
## O1  0.045  0.222  0.086 -0.016  0.588
## O2  0.241  0.068 -0.072  0.154 -0.595
## O3  0.060  0.290  0.028  0.075  0.637
## O4  0.221 -0.324 -0.015  0.268  0.491
## O5  0.139  0.081 -0.011 -0.003 -0.681
##
##                  PC2   PC1   PC3   PC5   PC4
## SS loadings    3.075 2.827 2.525 2.224 2.089
## Proportion Var 0.123 0.113 0.101 0.089 0.084
## Cumulative Var 0.123 0.236 0.337 0.426 0.510
```

. . . and the structure matrix shows us the correlations between the variables and the factors.

```
print(pc5.out$Structure, cutoff=0, digits=3)
```

```
##          PC2      PC1      PC3     PC5      PC4
## A1  0.212907  0.07575  0.02697 -0.5934 -0.12801
## A2 -0.035745  0.26336  0.20916  0.7440  0.11149
## A3 -0.060868  0.40052  0.18315  0.7285  0.10191
## A4 -0.115128  0.24698  0.30633  0.5475 -0.12221
## A5 -0.173505  0.49446  0.17643  0.6242  0.10010
## C1 -0.005854  0.11251  0.66075  0.0457  0.23987
## C2  0.070089  0.08830  0.73602  0.1510  0.12747
## C3 -0.046938  0.06560  0.67206  0.1578 -0.00462
## C4  0.317776 -0.12858 -0.69578 -0.0850 -0.14650
## C5  0.358435 -0.25977 -0.65150 -0.0870  0.02358
## E1  0.050010 -0.67013  0.00167 -0.1704 -0.11414
## E2  0.281251 -0.75686 -0.17694 -0.1844 -0.08279
## E3  0.003611  0.63889  0.16770  0.3479  0.32967
## E4 -0.132009  0.73416  0.18593  0.3744 -0.04613
## E5  0.022180  0.59806  0.41564  0.1630  0.26925
## N1  0.810704 -0.04224 -0.11145 -0.1713 -0.06896
## N2  0.801580 -0.07683 -0.09807 -0.1527 -0.00404
## N3  0.794194 -0.14576 -0.12412 -0.0072  0.01024
## N4  0.649684 -0.43258 -0.24507 -0.0138  0.06088
## N5  0.621782 -0.22946 -0.08086  0.1403 -0.17290
## O1  0.000476  0.28164  0.19364  0.0993  0.61646
## O2  0.237348  0.00047 -0.13944  0.0893 -0.58401
## O3  0.008558  0.35926  0.16591  0.2007  0.67314
## O4  0.252913 -0.25743 -0.01250  0.2333  0.48573
## O5  0.133315 -0.00152 -0.09460 -0.0601 -0.67603
```

Final Thoughts

Remember that factor analysis, despite being a mathematically intensive procedure, is highly subjective. You have to make choices about:

- the type of extraction method to use (principal axis or principal components)
- the number of factors to extract
- the factor rotation method to use when looking for simple structure (and interpretability)
- the interpretation of the factors

Furthermore. . . the outcome of your factor analysis (and the resulting interpretation of the factors) will be highly dependent upon the variables that you select for inclusion in the analysis. Factor analysis is an excellent method for evaluating underlying latent constructs, but. . . at it's core, it is simply a method of parsing large correlation matrices. If you select several variables that are highly related, you should not be surprised if they group together to form a factor!