Data Reduction and Index Construction

Joseph Nathan Cohen

Fall 2019

Contents

Introduction

Data reduction is the task of transforming a data set by reducing its number of variables. Such situations are more common in psychology, where a surveyor might use tens of questions to measure one respondent trait. The Minnesota Multiphasic Personality Inventory (a well-known psychopathology test) asks 567 questions. To make sense of this many variables, we have to boil them down to a smaller number. The human mind can only process so much complexity, and reduction allows people to make sense of an analysis involving many questions.

Of course, one way to reduce the number of variables in a set is to throw variables away. Alternatively, an analyst reduce the number of variables in a set by blending them. The analyst can combine multiple variables by boiling them down to one variable, synthesizing them into composite measures or "indexes". Here, ***numerical indexes*** are numerical scores that summarize the behavior of multiple metrics.

Some well-known indexes:

- The Dow Jones Industrial Average is as a metric that represents the behavior of the whole stock market by averaging the price performance of 30 stocks.
- The United Nations' Human Development Index (HDI) is a metric that measures socio-economic development based on countries' life expectancies, mean/expected years of schooling, amd gross national income per capita.
- *The Economist* publishes a Quality-of-Life index that ranks places to live by a range of material wellbeing, health, political, and social metrics.
- ESPN's MLB Relative Power Index rates the strength of Major League Baseball teams, based on a team's winning percentage, opponents' winning percentage, and opponents' opponents' average winning percentage.

Illustration

The methods that we will learn this week can be used to develop and validate efforts to consolidate and reduce variables in a data set. Figure 1 (below) presents a depiction of a hypothetical indexation operation.

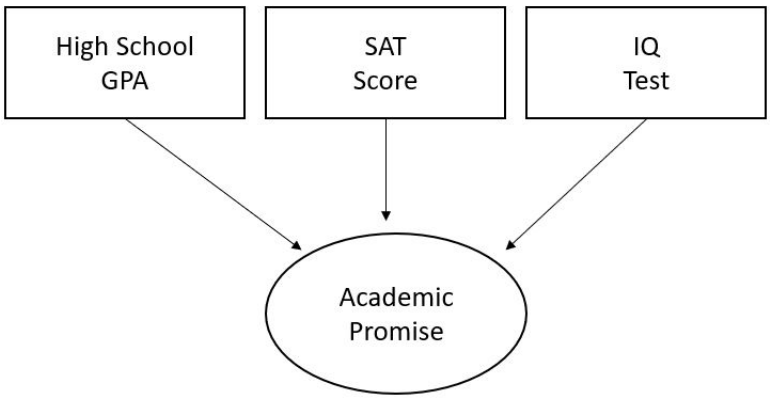


Figure 1: Depiction of a Hypothetical Data Reduction Operation

In this figure, we would describe "Academic Promise" as a *latent variable*, which is a variable that we cannot observe directly. It is latent – hiding beneath the surface. In contrast, GPA, SAT, and IQ scores are *observed variables* – we see these scores, because they are in our database.

The simple way to index these scores: (1) standardize each of these metrics (in order to put them on the same scale) and (2) take the average of the three standardized scores. However, doing so assumes that each of these three metrics are truly related to a common, underlying concept (academic promise), and are all equally important in measuring it. Moreover, this simple method assumes that you know how to measure "academic promise" in the first place – what if you have no idea? Factor analysis gives you a suite of tools to solve these kinds of problems.

We assume that our observed variables are related because of their association with a common underlying latent variable. In other words, we expect correlations between GPAs, SATs, and IQs because we presume that they are all outgrowths of one diffiult-to-observe quality: academic promise.

This Lesson

We discuss four methods for developing indexes:

- *Simple Indexation*, combining variables by averaging standardized variables
- *Cronbach's Alpha*, a quick-and-dirty (and commonly-used) method for justifying some combination of variables as related to an underlying latent variable.
- *Exploratory Factor Analysis*, which we use to search for common factors or latent variables in a larger number of variables.

Another method, *confirmatory Factor Analysis* is a more rigorous (and onerous) method for ascertaining whether or not we can infer that some latent variable underlies a collection of observed variables. We will bracket that to follow a lesson on *structural equation modeling*.

Data for this Lesson

In this module, we work two data sets. For our simple indexation lesson, we use data from the OECD *Better Life Index*, which can be downloaded here. The set measures 41 countries' overall quality-of-life using 24 different variables. You can learn more about the data set here.

For the remaining lessons, we use a generated data set that scores 400 respondents over 15 traits: happiness, optimism, sociability, anxiety, anger, jealousy, resentment, fear, boredom, tiredness, annoyance, irritability, hopefulness, friendliness, and ambition. These traits are all rated on a zero (inapplicable) to ten (fully applicable). We are interested in seeing whether we can reduce these 15 variables to a smaller number of factors. We are trying to see if we distill the major psychological states underlying these 15 variables. This set is called "Simulated Dispositional Data for EFA.xlsx", and can be downloaded here.

Both sets are in Excel format, and can be loaded using the **read_xlsx()** command in the ***readXL*** package. For example, to load the latter set:

```
library(readxl)
DATA <- read_xlsx("Simulated Emotional Disposition Data.xlsx", sheet = 1, col_names = TRUE)
```

Simple Indexes

We are studying different ways to consolidate variables in a data set. The simplest method for consolidating variables is to average standardized variable scores. However, this technique employs a range of assumptions that may not hold in reality. ‘ This method assumes:

- All variables are continuous
- All variables are capturing the same latent construct
- All variables reflect the underlying construct equally

The Model

$$s_{ij} = \frac{\sum_{=1}^{N} (\frac{x_{ij} - \bar{x_i}}{\sigma_i})}{N}$$

$$I = \frac{\sum_{i=1}^{M} s_{ij}}{M}$$

Where:

- I is the index score
- $x_i j$ is variable i for subject j
- $s_i j$ is a standardized transformation of $x_i j$ with a mean of zero and a standard deviation of one.
- $\bar{x_i}$ is the mean of variable x_i
- σ_i is the standard deviation of variable x_i
- N is the total number of subjects with reported values of x_i
- M is the total number of variables being used in the index

Implementation

Data

We are going to use the OECD *Better Life Index*[1], a database that seeks to measure 41 countries' living standards. You can download a copy of the data here

```
library(readxl)
DATA <- read_xlsx("OECD Better Life Data.xlsx", sheet = 1)
DATA <- data.frame(DATA)

#The set has 24 variables, which are explained in the dataset Excel workbook
#Here are the variable names:
names(DATA)
```

```
##  [1] "country"      "nofacilities" "houseexp"     "rooms"        "dispinc"
##  [6] "finwealth"    "labinsec"     "employment"   "ltunemp"      "earnings"
## [11] "support"      "edattain"     "skills"       "schoolyears"  "airpollution"
## [16] "waterqual"    "stakeeng"     "voters"       "lifeexp"      "selfhealth"
## [21] "satisfaction" "feelsafe"     "murder"       "longhours"    "leisuretime"
```

[1] OECD (2019) “OECD Better Life Index” Online database, available at http://www.oecdbetterlifeindex.org/

```
#A quick look at the top of the first few variables:
head(DATA[1:7], n=5)
```

```
##     country nofacilities houseexp rooms dispinc finwealth labinsec
## 1 Australia           NA       20    NA   32759    427064      5.4
## 2   Austria          0.9       21   1.6   33541    308325      3.5
## 3   Belgium          1.9       21   2.2   30364    386006      3.7
## 4    Canada          0.2       22   2.6   30854    423849      6.0
## 5     Chile          9.4       18   1.2      NA    100967      8.7
```

Step One: Standardize Variables

You begin by standardizing the variables that will be used in the index. Doing so puts all of the variables on the same scale, so that the index isn't unduly influenced by variables that are denominated in large numbers.

Below, we standardize the variables of our data set using the **standardize()** function in the ***psycho*** package.[2]:

```
library(psycho)
SDATA <- standardize(DATA)

#Rounding the data's numeric variables to two decimal places, just
#to make it easier to read:
SDATA[2:25] <- round(SDATA[2:25], 2)

#A look at the resulting set:
head(SDATA[1:7], n = 5)
```

```
##     country nofacilities houseexp rooms dispinc finwealth labinsec
## 1 Australia           NA    -0.26    NA    0.70      0.83    -0.24
## 2   Austria        -0.49     0.14 -0.04    0.81      0.11    -0.56
## 3   Belgium        -0.38     0.14  1.31    0.36      0.58    -0.53
## 4    Canada        -0.58     0.53  2.21    0.43      0.81    -0.14
## 5     Chile         0.51    -1.05 -0.94      NA     -1.14     0.32
```

Step Two: Average the Standardized Variables

The next step is to average the standardized variables. You may use mean or median. The former is perhaps more popular and arugably better incorporates the effect of each variable that comprises the index, but is also more likely to be influenced by outliers.

To average these variables, we use the **rowMeans()** command in the base package. We are asking the command to deliver the mean of scores in rows 2 to 25 in the SDATA data frame. The command's default is to render an "NA" if an observation has any missing values (i.e., it employs listwise deletion). We set the subcommand "na.rm = T" in order to get the average of all non-missing observations.

```
SDATA$wellbeing.1 <- rowMeans(SDATA[2:25], na.rm = T)

#Top 5 wellbeing using this consolidation strategy:
```

[2]We use this command in lieu of the base packages **scale()** command because **standardize()** ignores character variables instead of returning an error. This feature ends up being convenient when you are working with a set that has numeric and text variables interspersed throughout the data frame

```
library(dplyr)
arrange(SDATA,-wellbeing.1)[1:5,c(1,26)]
```

```
##         country wellbeing.1
## 1       Iceland   0.4809524
## 2   New Zealand   0.4740909
## 3     Australia   0.4618182
## 4   Switzerland   0.4542857
## 5 United States   0.4133333
```

```
#Bottom 5:
arrange(SDATA,wellbeing.1)[1:5,c(1,26)]
```

```
##    country wellbeing.1
## 1    Chile  -0.6300000
## 2  Hungary  -0.5472727
## 3   Russia  -0.5294444
## 4 Colombia  -0.4783333
## 5   Brazil  -0.4740000
```

Pitfalls

Before enumerating the pitfalls associated with the simple index method of consolidating variables, readers might have noted that missing data is a problem in this analysis. Of the 40 countries in this data set, only 15 have data on all 24 variables. Another 20 are missing 1 - 3 values, and five more are missing five or more values. The result is that c ountreis' "wellbeing" scores are generated by an inconsistent mix of variables. This is not a weakness of simple indexes, but a generic missing data problem. Were someone to approach me with this data and concerns about missing data, I would recommend using multiple imputaton with randomness, which we bracket here but discuss in another module.

Now, let's turn to problems that are more specific to simple indexation as a data consolidation method. I can think of a few pitfalls associated with this indexation method offhand, which are related to this indexation method's underlying assumptions. The fragility of these assumptions are easier to see when we consider the indexation operation above.

In my view, the biggest pitfall with this method is that it can tempt anaylsts to build indexes mechanistically without thinking about whether their data consolidation strategy makes sense. Consider the "wellbeing" index that we build using OECD data in the above example. We built a "wellbeing" index using the mean of 24 standardized variables. In so doing, we assumed that:

- All 24 variables were continuous (which they were!)
- All 24 variables were measuring the same thing
- Each of our 24 variables captured precisely 1/24th of countries' "wellbeing" levels

In other words, the method assumes that people's subjective feelings of safety are just as important as their children's performance on standardized aptitude tests, each of which are precisely as important as water quality. Moreover, the data has three variables related to housing, but only two related to physical safety. If we simply calculate the mean across all our variables, we are implicitly assuming that housing is 50% more important than physical safety in shaping a country's overall wellbeing.

There are easy statistical fixes for dealing with this kind of situation. For example, the first situation – concerns that these individual wellbeing metrics aren't equally important can be address by weighting. For

example, I could manually specify weights associated with each of this set's 24 variables. In the example below, I double the importance of the public safety-related metrics, and multiply the health metrics' importance by 10. We then calculate an alternative wellbeing index using the **rowWeightedMeans()** command from the ***matrixStats*** package:

```
#We are going to use variables 2 to 25 - 24 total variables
#We specify weights for each of these 24, as they appear in the set.
#See the results of the names(DATA) operation above.

weights <- c(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,10,10,1,3,3,1,1)

#To double check the weights you are assigning to each variable:
#cbind(names(SDATA[2:25]), weights)

#Recalculate the Wellbeing index:
library(matrixStats)
```

```
##
## Attaching package: 'matrixStats'

## The following object is masked from 'package:dplyr':
##
##     count
```

```
SDATA$wellbeing.2 <- rowWeightedMeans(as.matrix(SDATA[2:25]), weights, na.rm = T)

#Does it affect our rankings? Top 5: (I swear this is a coincidence)
arrange(SDATA,-wellbeing.2)[1:5,c(1,26, 27)]
```

```
##       country wellbeing.1 wellbeing.2
## 1      Canada   0.4037500   0.6263043
## 2 New Zealand   0.4740909   0.6081818
## 3 Switzerland   0.4542857   0.6051163
## 4   Australia   0.4618182   0.5852273
## 5     Iceland   0.4809524   0.5279070
```

```
#Bottom 5:
arrange(SDATA,wellbeing.2)[1:5,c(1,26,27)]
```

```
##        country wellbeing.1 wellbeing.2
## 1 South Africa  -0.2620000  -1.7053571
## 2       Russia  -0.5294444  -0.9940000
## 3    Lithuania  -0.3590476  -0.7793023
## 4       Latvia  -0.4166667  -0.7015217
## 5      Hungary  -0.5472727  -0.5750000
```

Insofar as concerns that different domains of wellbeing (e.g., health, safety, economic life) have different numbers of variables in this set (see above and the set's codebook). There are a lot of economic variables, which is part of the reason that the US performs so wellin the first measure but less well in the second (US living standards are strongly buoyed by their high levels of material wealth). Many well-known indexes of this type deal with this concern by consolidating variables into first-order indexes that group empirical

variables according to some common theme, and then calculate a higher-order, overall index by averaging these first order indexes.

For example, we might create a Health index using this OECD data as follows. And then we might repeat the same procedure with the other domains of wellbeing described in the OECD set.

```
SDATA$health <- rowMeans(SDATA[c(20,21)])
```

But ultimately we are always left with the questions surrounding our assumption that these variables are actually related to the same underlying construct "wellbeing". We are just assuming that there is a thing called "wellbeing", and that it is somehow made up of things like work-life balance, safety, housing, and the other major categories described in the set. How do we know this is true?

A more empirically-minded analyst might want to check the data for evidence that these variables are in fact measure what we think they are measuring. These are the subjects of this module's other three lessons.

Cronbach's Alpha

Cronbach's Alpha tests of the assumption that a set of variables reliably predict a common underlying concept. It tests whether there are strong relationships among variables that purport to measure the same underlying construct. If this tests fails, then you might question whether that variable set does in fact measure the same thing.

Model

We can calculate Cronbach's Alpha via our variab'es correlations, where:[3]

$$\alpha = \frac{N \times \bar{c}}{v + (N-1) \cdot \bar{c}}$$

Where:

- α is the Cronbach's Alpha score
- N is the number of variables being tested
- $\bar{c}$ is the average of covariance between variables
- $\bar{v}$ is the average variance of each variable

Roughly, the metric approximates the ratio of inter-item covariance to our variables' overall variance.

Implementation

Data

Our data is this module's simulated emotional disposition data, contained in the Excel spreadsheet "Simulated Emotional Disposition Data for EFA.xlsx". The set scores 400 respondents over 15 traits: happiness, optimism, sociability, anxiety, anger, jealousy, resentment, fear, boredom, tiredness, annoyance, irritability, hopefulness, friendliness, and ambition. These traits are all rated on a zero (inapplicable) to ten (fully applicable).

[3] following Goforth, Chelsea (2015) "Using and Interpreting Chronbach's Alpha" University of Virginia Library Research Data Services https://data.library.virginia.edu/using-and-interpreting-cronbachs-alpha/

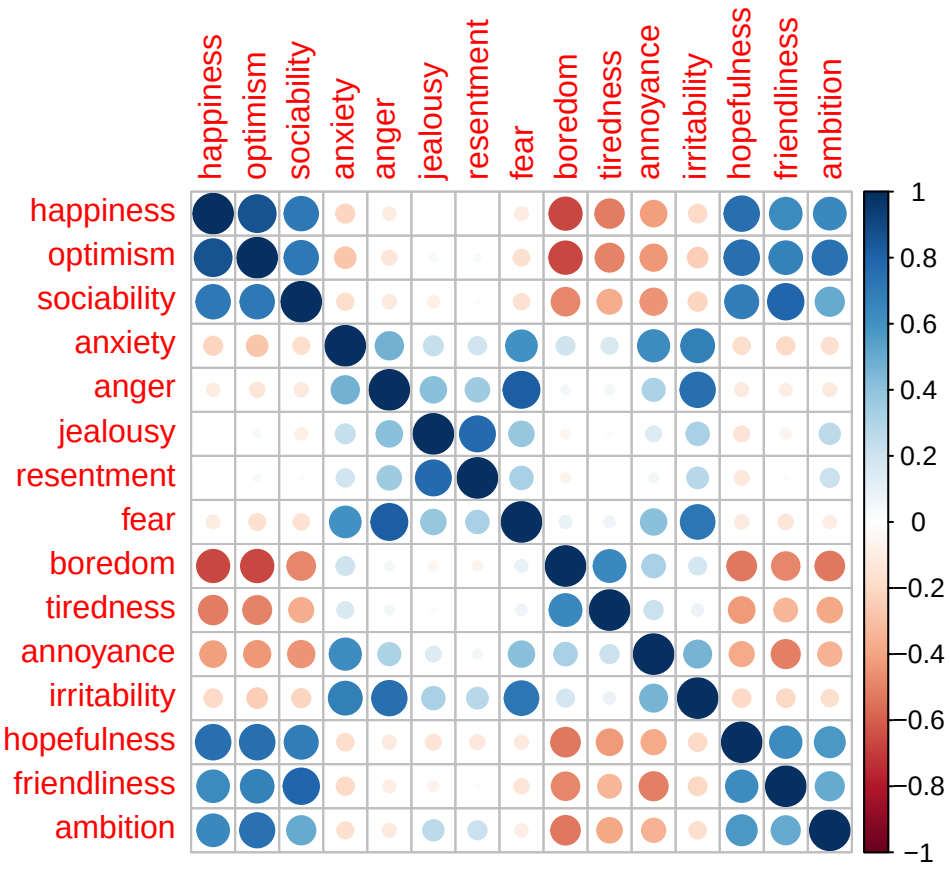
```
library(readxl)
DATA <- read_xlsx("Simulated Emotional Disposition Data.xlsx", sheet = 1)
DATA <- data.frame(DATA)

#Rounding the data's scores to make save space in these printouts:
DATA[2:16] <- round(DATA[2:16], 2)

#A peek at the first few data points in the set:
head(DATA[1:7], 5)
```

```
##   id happiness optimism sociability anxiety anger jealousy
## 1  1         6        7           7       5     5        5
## 2  2         7        8           7       3     8        6
## 3  3         5        5           6       3     5        7
## 4  4         3        3           3       5     4        5
## 5  5         6        6           8       7     6        6
```

Step One: Calculate a Correlation Matrix

Our first step is to create a matrix of correlations among the items we are trying to index. If our data is continuous, we can calculate these correlations using the **cor()** command in the base package. If you are working with binary or ordinal data, use the **polychoric()** command in the ***psych*** package.

```
#We are using variables 2 - 16 in this data set.
#They are all continuous, so we use cor().
#Let's start off by finding highly related metrics:
CORRS <- cor(DATA[2:16])

#Visualizing using corrplot (see next page)
library(corrplot)
corrplot(CORRS)
```



The results suggest a very strong relationship between happiness, optimism, sociability, hopefulness, friendliness, and ambition. We construct a correlation matrix with only these variables:

```
#Calculate correlation matrix with variables we are going to index
CORRS.2 <- cor(DATA[c(2:4,14:16)])

#A peek at the top of the matrix:
round(CORRS.2, 2)[,1:5]
```

```
##              happiness optimism sociability hopefulness friendliness
## happiness         1.00     0.86        0.71        0.76         0.64
## optimism          0.86     1.00        0.71        0.76         0.68
## sociability       0.71     0.71        1.00        0.69         0.80
## hopefulness       0.76     0.76        0.69        1.00         0.63
## friendliness      0.64     0.68        0.80        0.63         1.00
## ambition          0.64     0.74        0.51        0.57         0.50
```

Remember that the variables you are testing should have positive correlations. If this happens, reverse-code and reconceptualize the variable you are testing. For example, *happiness* and *boredom* share a strong, negative relationship. If I wanted to measure their association in a Cronbach's Alpha test, I might do something like reverse-code *boredom* and call it something like *engagedness*.

Step Two: Calculate Cronbach's Alpha

We calculate a Cronbach's Alpha using the **alpha()** command in the ***psych*** package:

```r
library(psych)
alpha <- alpha(CORRS.2)

#Results:
alpha
```

```
##
## Reliability analysis
## Call: alpha(x = CORRS.2)
##
##   raw_alpha std.alpha G6(smc) average_r S/N median_r
##       0.93      0.93    0.93      0.68  13     0.69
##
##  Reliability if an item is dropped:
##              raw_alpha std.alpha G6(smc) average_r  S/N  var.r med.r
## happiness         0.91      0.91    0.91      0.66  9.7 0.0106  0.69
## optimism          0.90      0.90    0.90      0.65  9.1 0.0097  0.64
## sociability       0.91      0.91    0.91      0.68 10.5 0.0109  0.66
## hopefulness       0.91      0.91    0.92      0.68 10.6 0.0130  0.70
## friendliness      0.92      0.92    0.91      0.70 11.5 0.0101  0.71
## ambition          0.93      0.93    0.93      0.72 13.1 0.0053  0.71
##
##  Item statistics
##                 r r.cor r.drop
## happiness    0.90  0.88   0.85
## optimism     0.93  0.92   0.89
## sociability  0.86  0.84   0.80
## hopefulness  0.86  0.82   0.79
## friendliness 0.83  0.79   0.75
## ambition     0.77  0.71   0.67
```

Step Three: Interpret the Results

Key elements of the output:

- **raw_alpha** return above gives Cronbach's Alpha based on covariances.

- **std.alpha** is a standardized alpha based on correlations.
- **average_r** is the average correlation between the variables
- **median_r** is the median correlation among these variables

The lower panel of the output gives you a sense of how our results would change if we were to drop one of the items in our variable set.

Focus on **raw_alpha**. In this case, the score is 0.93.

- A score about 0.80 is conventionally treated as sufficient in the peer-reviewed literature.
- Some methodologists argue that the threshold could be lowered to 0.70, though doing so injects anti-conservatism into your analysis.
- I have also heard the argument that an alpha over ~0.95 could be a sign that your analysis is relying on redudant variables, rather than multiple metrics that triangulate the measurement of a common underlying construct.

###Step Four: Score the Index

Once you have found a grouping that your *alpha* says operationalizes a common underlying construct, you can agglomerate them as a standardized index, as we did above.

Exploratory Factor Analysis

Exploratory Factor Analysis (EFA) is a method that uses correlations or covariance metrics to reduce a larger number of variables to a smaller number of latent variables (or "factors"). This process is sometimes called "reducing the dimensionality" of a data set. We will perform these operations using the **fa()** function in the ***psych*** package.

Illustrative Example

Survey researchers ask a battery of 50 political opinion questions. They suspect that many of these opinions are related, and that these 50 questions could be boiled down to fewer "meta-issues" or dimensions of political opinion. For example, we might find that attitudes towards taxes, government regulation, and free trade might be correlated such that we could reduce people's answers on these three questions to one "free market-government intervention" scale. Alternatively, we might be able to combine bleiefs about abortion, school prayer, and public funding of religious schools down to a metric on attitudes towards "religion". You would use an exploratory factor analysis to identify these kinds of variable clusters empirically.

Guidelines

Some guidelines:[4]

- The method assumed univariate and multivariate normality in data,
- It assumes a linear relationship between factors and variables,
- Factors should have at least three variables
- Recommended sample size of N $>$ 300
- Ratio of respondents to variables should be at least 10:1, but more than 30:1 is preferable
- Correlations should be greater than 0.30 among variables that are potentially related to a common factor
- Variables related to teh same factor should be positively correlated, so reverse-code variables with strong negative correlations associated with the same factor.

The Model

EFA is based on regressions, but predicts individual variable values as a function of underlying latent variables (or "factors"):

$$X_j = a_{j1} \cdot F_1 + a_{j2} \cdot F_2 + ... + a_{jm} \cdot F_m + \epsilon_j$$

In which:

- X_j is variable j to be expressed as a sum of factors F_m
- F_m is factor m

[4]Source: An Gie Yong and Sean Pierce (2013) "A Beginner's Guide to Exploratory Factor Analysis" *Tutorials in Quantitative Methods for Psychology*, 9 (2): 79 - 94.

- a_{jm} is factor loading m for variable j
- ϵ_j is an error term

An extended exposition of the method is given in Everitt and Hothern (2011, Ch. 5).[5] In effect, the procedure uses variable correlations as a basis for estimating the latent factors' coefficients (or "factor loadings") in the system of equations above.

Implementation

Data

Our data is this module's simulated emotional disposition data, contained in the Excel spreadsheet "Simulated Emotional Disposition Data.xlsx". The set scores 400 respondents over 15 traits: happiness, optimism, sociability, anxiety, anger, jealousy, resentment, fear, boredom, tiredness, annoyance, irritability, hopefulness, friendliness, and ambition. These traits are all rated on a scale of zero (fully inapplicable) to ten (fully applicable).

```
library(readxl)
DATA <- read_xlsx("Simulated Emotional Disposition Data.xlsx", sheet = 1)

#Convert DATA object to data frame:
DATA <- data.frame(DATA)

#A quick look at the data (table is truncated for legibility):
head(DATA[1:8], 5)
```

```
##   id happiness optimism sociability anxiety anger jealousy resentment
## 1  1         6        7           7       5     5        5          6
## 2  2         7        8           7       3     8        6          5
## 3  3         5        5           6       3     5        7          5
## 4  4         3        3           3       5     4        5          4
## 5  5         6        6           8       7     6        6          4
```

From the outset, it is important to stress that we are working with simulated data (click here for the script used to generate the data). These data have been designed to create strong results. You would be hard pressed to find results that work this well in the real world. Keep that in mind when you conduct your own analyses and are tempted to use these findings as a benchmark against which to compare your own results.

Step 1: Standardize Variables

This is a step to take if you are trying to reduce variables that are denominated on different scales. This is not the case here, as all of our variables are denominated on the aforementioned zero to ten scale.

If this were not the case, and we wanted to standardize our continuous variables so that they are denominated on a common scale, we would use the **standardize()** function in the ***psycho*** package.[6]

[5] Brian Everitt and Torsten Hothorn (2011) *An Introduction to Applied Multivariate Analysis with R* Springer.

[6] We use this command in lieu of the base packages **scale()** command because **standardize()** ignores character variables instead of returning an error. This feature ends up being convenient when you are working with a set that has numeric and text variables interspersed throughout the data frame

```
library(psycho)
STD.DATA <- standardize(DATA[2:16])

#A quick look at the standardized data (table is truncated
#and figures are rounded for legibility):
head(round(STD.DATA[1:8],1), 3)
```

```
##   happiness optimism sociability anxiety anger jealousy resentment fear
## 1       0.6      1.0         0.9     0.0   0.1     -0.1        0.8 -0.6
## 2       1.1      1.4         0.9    -1.2   1.5      0.7       -0.1 -0.2
## 3       0.1      0.0         0.4    -1.2   0.1      1.5       -0.1  0.6
```

Step 2: Calculate correlation or covariance matrix for EFA

Next, calculate the correlation or covariance matrix of the variables you wish to use in the EFA. If all of your variables are continuous, then you can use the **cor()** function in the base package. That is what we will do here.

However, if your EFA includes nominal or ordinal variables, use **polychoric()** in the ***psych*** package to get polychoric correlations.

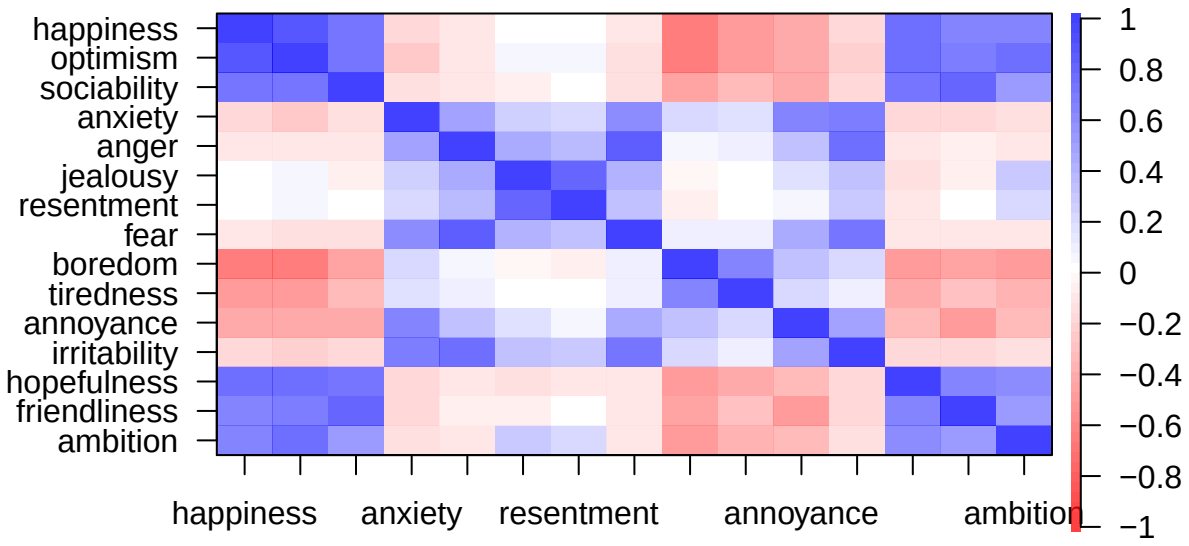
Note that, in the operation below, I do not use the first column of the data frame. That column contains unit identifiers.

```
# Run the operation to obtain polychoric correlations of the
# variables we wish to run in the EFA
CORR <- cor(DATA[-1])

# Isolate the number of observations used in calculating the correlations.
# You will use this in the next step.
N.OBS <- nrow(DATA)

#Here's the first few columns of the resulting matrix.
#It is rounded to the second decimal place to improve legibility
round(CORR[,c(1:6)], 2)
```

```
##              happiness optimism sociability anxiety anger jealousy
## happiness         1.00     0.86        0.71   -0.21 -0.10    -0.01
## optimism          0.86     1.00        0.71   -0.27 -0.13     0.03
## sociability       0.71     0.71        1.00   -0.17 -0.12    -0.08
## anxiety          -0.21    -0.27       -0.17    1.00  0.48     0.23
## anger            -0.10    -0.13       -0.12    0.48  1.00     0.42
## jealousy         -0.01     0.03       -0.08    0.23  0.42     1.00
## resentment        0.00     0.03       -0.02    0.19  0.35     0.78
## fear             -0.11    -0.16       -0.16    0.60  0.82     0.38
## boredom          -0.66    -0.67       -0.49    0.20  0.05    -0.06
## tiredness        -0.51    -0.50       -0.37    0.16  0.06     0.02
## annoyance        -0.42    -0.44       -0.44    0.62  0.31     0.14
## irritability     -0.20    -0.25       -0.21    0.69  0.75     0.32
## hopefulness       0.76     0.76        0.69   -0.18 -0.11    -0.15
## friendliness      0.64     0.68        0.80   -0.19 -0.09    -0.07
## ambition          0.64     0.74        0.51   -0.17 -0.11     0.26
```

You can visualize the correlation matrix using **cor.plot()** in the ***psych*** package.

```
library(psych)
cor.plot(CORR)
```



Step 3: Run an EFA

Then, use the **fa()** command with the resulting correlation matrix. Here is an example of the implementation. An explanation of your options follows:

```
#Run the EFA using the correlation matrix
library(psych)
EFA.MODEL <- fa(CORR,
                nfactors = 3,        #Specifies number of factors
                n.obs = N.OBS,       #Number of observations
                fm = 'minres',       #Factoring method
                rotate = 'varimax')  #Specify rotation method
```

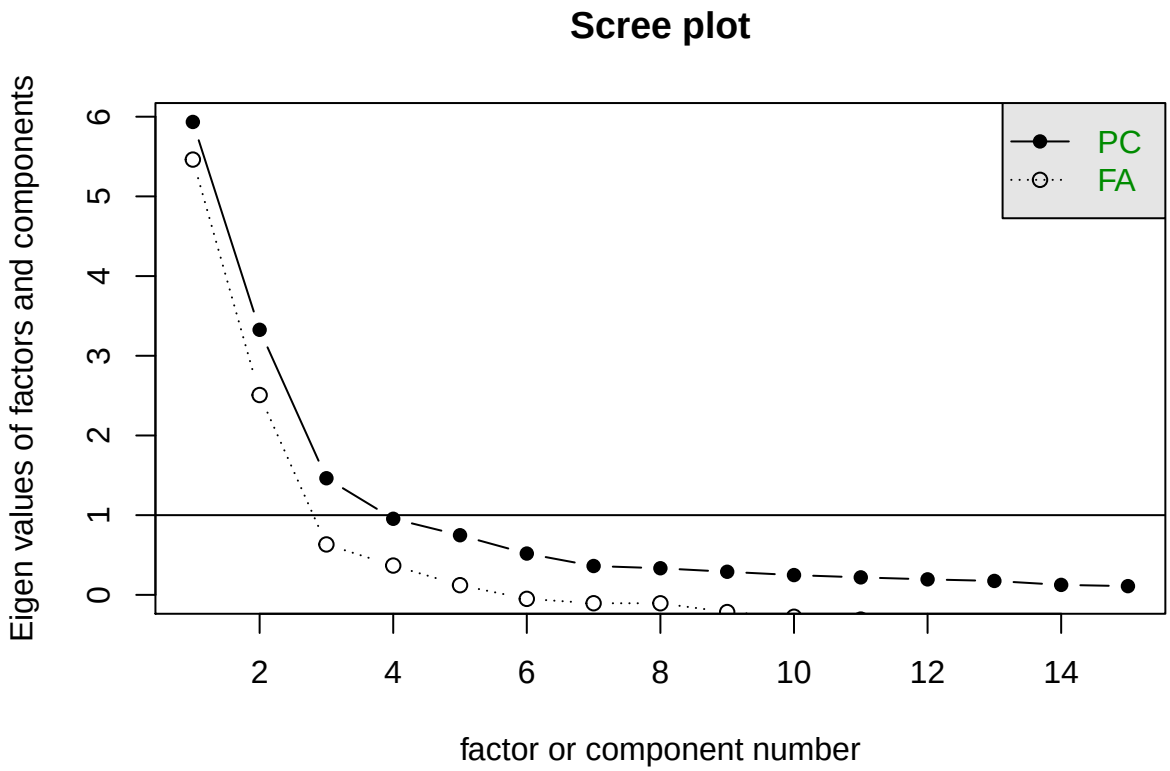
This operation includes three choices: the number of factors, the factoring method, and the rotation method.

3.1: Factoring Method *Factoring method* (chosen by the "fm=" option) specifies the method by which we calculate factors. The methods are discussed in greater detail in Everett and Hothorn. Here, I used the 'minres' option, which relis on OLS regression. This is a default that works well in most situations. A 'gls' is said to increase the impact of unique/uncorrelated variables in your result. An "ml" specification

runs a maximium-likelihood analysis, which many audiences will perceive as "more respectable" according to Everett and Hothorn

3.2: Number of Factors *Number of factors* (chosen by the "nfactors=" option) specifies the number of factors to model. The goal is to model enough factors to capture as much variation in the data is possible, but without additional superfluous factors that capture minimal variation. One way to depict these diminishing returns is by generating a scree plot using your correlation matrix. You can do this with the **scree()** command:

```
scree(CORR)
```



We are looking at the "FA" line in this figure. The "PC" is for a principal components analysis – another operation. The vertical axis can be understood as depicting the variance absorbed by each additional factor added to the model. It suggests that our model absorbs substantially more variance in adding a second or third factor, but that the additional gains when we add a fourth or more factors are minimal. This inflection point at M = 3 suggests that using three factors gives us a balance between explaining variance and parsimony.

3.3: Rotation Method Factor rotation involves transforming the results of the analysis to make interpreting the factors easier. Here's a handy guide to rotation methods from IBM:

- *Varimax Method.* An orthogonal rotation method that minimizes the number of variables that have high loadings on each factor. This method simplifies the interpretation of the factors.

- *Direct Oblimin Method. A method for oblique (nonorthogonal) rotation. When delta equals 0 (the default), solutions are most oblique. As delta becomes more negative, the factors become less oblique. To override the default delta of 0, enter a number less than or equal to 0.8.
- *Quartimax Method.* A rotation method that minimizes the number of factors needed to explain each variable. This method simplifies the interpretation of the observed variables.
- *Equamax Method.* A rotation method that is a combination of the varimax method, which simplifies the factors, and the quartimax method, which simplifies the variables. The number of variables that load highly on a factor and the number of factors needed to explain a variable are minimized.
- *Promax Rotation.* An oblique rotation, which allows factors to be correlated. This rotation can be calculated more quickly than a direct oblimin rotation, so it is useful for large datasets.

Step 4: Adjust for Factor Loadings

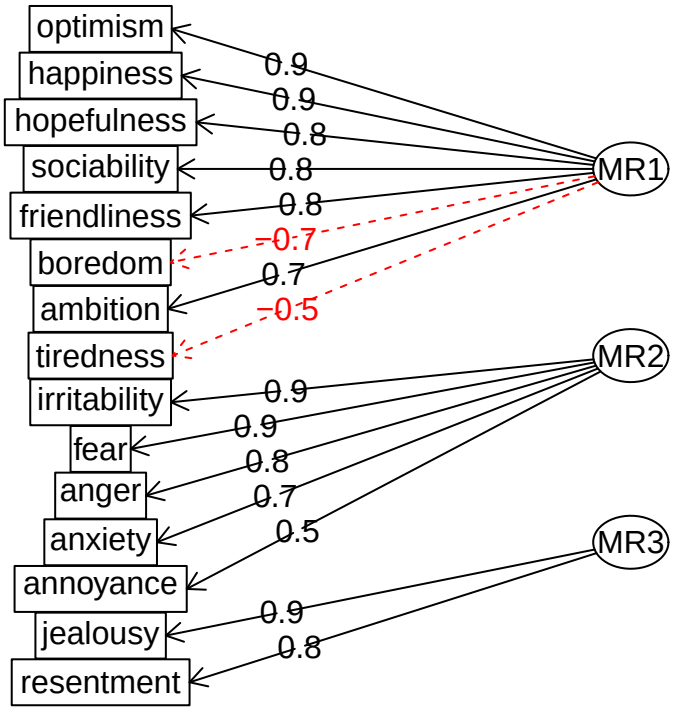
Factor loadings are describe the relationship between variables and factors. They are like coefficients. The resulting factor loadings are stored as a sub-object:

```
EFA.MODEL$loadings
```

```
##
## Loadings:
##              MR1    MR2    MR3
## happiness     0.898
## optimism      0.916 -0.156
## sociability   0.799 -0.103
## anxiety      -0.174  0.724
## anger                0.790  0.242
## jealousy             0.248  0.939
## resentment           0.209  0.768
## fear                 0.853  0.195
## boredom      -0.704
## tiredness    -0.549
## annoyance    -0.437  0.505
## irritability -0.133  0.863  0.125
## hopefulness   0.820        -0.157
## friendliness  0.756 -0.117
## ambition      0.699 -0.141  0.294
##
##                  MR1   MR2   MR3
## SS loadings    5.056 3.076 1.718
## Proportion Var 0.337 0.205 0.115
## Cumulative Var 0.337 0.542 0.657
```

Factor loadings can be visualized using the command **fa.diagram()**:

```
fa.diagram(EFA.MODEL)
```



Ideally, there will be either strong (e.g., >0.80) or low (near-zero) loadings. The variables with high factor loadings can be used to make sense of the factor's meaning. For example, in the figure above, Facotr M2 might be characterized as "visceral negative emotional reations" – at least that's one idea for tying together these related variables. Factor analysis involves some interpretive work, not just math.

Conventionally, for a sample size of N > 300, a rotated loading should be at least 0.32 to be considered substantial. Things can get more complicated when you have variables that *crossload* (with loadings of at least 0.32 on at least two factors). Our "annoyance" variable is cross-loading here (see table above). I'm going to respond by removing it from the analysis.

Also, you should examine the variables to ensure that none of the factor loadings register as negative. If they do, you should reverse-code the variable so that factor loadings are all positive. It looks like I need to reverse-code "tiredness" and "boredom"

Finally, it looks like "jealousy" and "resentment" are grouped in a factor of their own. Recall that it is recommended that a factor should have at least three variables that primarily load onto it. For this and other reasons that emerged in this analysis, I'm going to eliminate these variables.

I perform these adjustments here:

```
DATA.2 <- DATA

#Reverse coding variables:
DATA.2$tiredness <- -1 * DATA.2$tiredness
DATA.2$boredom <- -1 * DATA.2$boredom

#A new correlation matrix without columns of
#variables that I plan to exclude (and with identifier excluded):
```

```r
DATA.2.t <- DATA.2[-c(1,7,8,12)]
CORR.2 <- cor(DATA.2.t)

#Rerun the analysis, now with 2 factors:
EFA.MODEL.2 <- fa(CORR.2,
                  nfactors = 2,        #Specifies number of factors
                  n.obs = N.OBS,       #Number of observations
                  fm = 'minres',       #Factoring method
                  rotate = 'varimax')  #Specify rotation method
```

Step 5: Interpret the Results

5.1: Factor Loadings The operation returns an object that delivers a lot of information. Start by examining the factor loadings, as in Step 4. Look for cross-loaded variables, orphaned variables, factors with only one or two variables primarily loaded onto it, and whatever other problems you might detect.

```r
EFA.MODEL.2$loadings
```

```
##
## Loadings:
##              MR1    MR2
## happiness     0.902
## optimism      0.927 -0.145
## sociability   0.787 -0.119
## anxiety      -0.169  0.661
## anger                0.853
## fear                 0.892
## boredom       0.714
## tiredness     0.556
## irritability -0.135  0.876
## hopefulness   0.806
## friendliness  0.742 -0.112
## ambition      0.705
##
##                   MR1   MR2
## SS loadings     4.857 2.811
## Proportion Var 0.405 0.234
## Cumulative Var 0.405 0.639
```

It can help to see them visualized, using **fa.diagram()** as above. Here, the analysis seems to have performed fine. Pretty strong loadings, no cross-loadings, an sensible groupings.

Below the factor loadings, you will see sum-of-square estimates. This describes the proportion of variable caputred by each factor. Factor MR1 captures about 41% of variance, and MR2 caputres another 23%. Jointly, these factors capture 64% of variance.

5.2: Communiality & Uniqueness Estimates *Communality* of variables represent the variance that each variable shares with others via the common factors. *Uniqueness* is the proportion of each variable's variation not attributed to the factor. Both estimates should sum to 100%.

```
EFA.MODEL.2$communalities
```

```
##    happiness     optimism  sociability      anxiety        anger         fear
##    0.8216510    0.8801044    0.6343575    0.4650925    0.7281047    0.7968404
##      boredom    tiredness irritability  hopefulness friendliness     ambition
##    0.5159365    0.3122122    0.7858441    0.6598850    0.5631188    0.5050897
```

```
EFA.MODEL.2$uniquenesses
```

```
##    happiness     optimism  sociability      anxiety        anger         fear
##    0.1783500    0.1198960    0.3656421    0.5349090    0.2718943    0.2031558
##      boredom    tiredness irritability  hopefulness friendliness     ambition
##    0.4840638    0.6877868    0.2141581    0.3401162    0.4368811    0.4949104
```

Consider the "optimism" variable. These results suggest that about 88% of the observed variation in "optimism" scores are a product of the underlying factor that drives other variables like "happiness" or "sociability". If we reduce this variable into a a factor with these other variables, we aren't likely to lose as much informatino.
On the other hand, "tiredness" has relatively more uniqueness (0.68). Our factors are not aborbing a lot of the observed variation here.

5.3: Complexity Hoffman's Complexity Index gives the average number of factor needed to account for our observed variables variables.[7] A factor analysis with a simple structure will allow us to group like variables into single factors, in which case it would only require about one factor to summarize it. Recall that we removed cross-loaded factors, which required more than one factor to describe their variance. If you were to run complexity metrics on those variables, you'd find that the need about 1.5 factors.

```
EFA.MODEL.2$complexity
```

```
##    happiness     optimism  sociability      anxiety        anger         fear
##     1.020436     1.049221     1.045871     1.130488     1.000487     1.003389
##      boredom    tiredness irritability  hopefulness friendliness     ambition
##     1.025960     1.016915     1.047792     1.030124     1.045453     1.034794
```

5.4: Fit Statistics The operation returns several fit statistics.

```
summary(EFA.MODEL.2)
```

```
##
## Factor analysis with Call: fa(r = CORR.2, nfactors = 2, n.obs = N.OBS, rotate = "varimax",
##     fm = "minres")
##
## Test of the hypothesis that 2 factors are sufficient.
## The degrees of freedom for the model is 43  and the objective function was  1.15
## The number of observations was  400  with Chi Square =  451.66  with prob <  1.5e-69
##
## The root mean square of the residuals (RMSA) is  0.05
```

[7] Erik Pettersson and Eric Turkheimer (2010) "Item Selection, Evaluation, and Simple Structure in Personality Data" *Journal of Research in Personality*, 44: 407 - 420.

```
## The df corrected root mean square of the residuals is  0.06
##
## Tucker Lewis Index of factoring reliability =  0.825
## RMSEA index =  0.156  and the 10 % confidence intervals are  0.142 0.167
## BIC =  194.03
```

The results return four clusters of information. Many of these metrics are more important in confirmatory factor analysis, but they can still be information sources:[8]

- The first cluster, comprised of the top line, repeats the specification of the model being summariezed
- The second cluster, a three line set, gives a chi-square test of the null hypothesis that our model fits perfectly. You want a p-value above 0.05. Rejecting the null implies that your system of factors are not fully capturing the variance in your variable set.
- The bottom cluster includes several fit statistics. ** The Tucker-Lewis Index (TLI) gives the fit improvement over the null model. In confirmatory factor analysis (CFA), one would strive for a score of at least 0.95 ** The Root Mean Squared Error of Approximation (RMSEA) is a parsimony-adjusted index, and one strives for a score below 0.08 in CFA ** The Bayesian Information Criterion (BIC) is a fit statistic. An in-depth discussion of this criterion can be read here. Use this to compare different attempts at factor analysis. The lower BIC score is the better-fitting model.

Overall, this is a poor-fitting system of factors. However, the exercise did give us leads about which kinds of variables might be combined to reduce the dimensionality of our set. To get a more definitive answer on whether the kinds of data reductions we see here are doable, the gold standard method is confirmatory factor analysis

Step 6: Scoring Your Factors

How to convert these factors into data? The process is not straightforward, and different options are available. We will use the operation **factor.scores()** from the package ***psych***.

```
SCORES.2  <- factor.scores(DATA.2.t, EFA.MODEL.2, method = "Thurstone")

#Factor scores for two factors:
head(SCORES.2$scores, 10)
```

```
##               MR1        MR2
##  [1,]  0.85185068 -0.1737087
##  [2,]  1.45538437  0.6110634
##  [3,]  0.06769208  0.2840449
##  [4,] -1.04910526 -0.5637154
##  [5,]  0.77116265  0.8856597
##  [6,] -1.76064770  0.7130632
##  [7,]  1.50167781 -1.2151657
##  [8,]  0.38002375  1.3160078
##  [9,] -0.58469065  0.6312746
## [10,]  0.43540632  2.1098342
```

```
SCORES.DAT <- SCORES.2$scores

#Adding scores to data frame:
DATA.2 <- cbind(DATA.2, SCORES.DAT)
```

[8] For more on CFA fit statistics, our source is this useful handout from Cornell Statistical Consulting