Q-STEP R 'HOW TO' GUIDES:

ROBUSTNESS CHECKS WITH SURVEY RESEARCH 3: EXPLORATORY AND CONFIRMATORY FACTOR ANALYSIS

Creator: Dr James Weinberg

In most social science disciplines, quantitative researchers will work with survey research to develop, build or test theories. In order for such survey research to be as rigorous as possible, it is important that we conduct robustness checks on our results in order to assess the validity of our theoretical claims. Therefore, this guide (the last in a series of 3) focuses on exploratory and confirmatory factor analysis with questionnaire items.

In this guide, you will be given a simple contextual description of each type of factor analysis and when/why it should be used, as well as examples worked through in Rstudio. This guide assumes a basic competency in R from the start - for example users should already be comfortable with assigning and calling objects.

The example used in this guide is based on a dataset of people's Basic Human Values. Basic values are a personality characteristic that can be measured by psychometric surveys. In this instance, a 20 item questionnaire was administered to 107 people, with two items each tapping one of the ten basic values in the theory. These ten values can be clustered further into 4 higher order values. This guide will use exploratory and confirmatory factor analyses to assess the reliability of the questionnaire items used to test this theory.

.

Q-Step
A step-change in quantitative social science skills
Funded by the Nuffield Foundation, ESRC and HEFCE
Nuffield Foundation
Sheffield Methods Institute.
The University Of Sheffield.

Section 1

Exploratory Factor Analysis

What is it and when to use it?

Studies in the social sciences are often involve long lists of variables that are used in tandem to characterise other, unobservable, objects. An obvious example of this can be found in survey research, where questionnaires are employed that have lots of different questions in them. The number of variables created by these lists of questions can cause complications when it comes to interpreting and analysing the data, especially if we hope and/or assume that these questions are measuring the same (or a number) of other, underlying variables.

In these situations we can use exploratory factor analysis (EFA) to group variables that are intercorrelated under more general latent factors. The overall purpose of EFA is to reduce the original list of variables to a smaller number of common factors that are easily interpretable. Factor analysis therefore provides a clearer view of the data AND new parsimonious variables to use in ongoing analyses.

Each new factor creates a dimension that can understood as a classification axis along which our measurement variables are plotted. This projection produces two sets of results known as factor scores and factor loadings. Factor scores are "the scores of a subject on a [...] factor" (Rietveld & Van Hout 1993: 292), while factor loadings are the "correlation of the original variable with a factor" (ibid: 292). We can then use the factor scores for future tests like regression models, while the factor loading allows us to determine the importance of each measurement variable for that factor (as a correlation, it can be squared to provide a figure for the variance accounted for by the variable in question). We can also use the data to interpret the new factors and give them theoretically sensible labels.

Example

Start by setting your working directory and reading the data file containing your questionnaire responses. You can conduct EFA using the "psych" package.

It is likely that your survey contains a lot more items than you need for this analysis (i.e. socio-demographic data or another item battery). For example, in my dataset I have 165 variables but for the purpose of the current test, I am only interested in the 20 question battery related to respondents' basic values. Therefore, you need to isolate these data as new matrix in your global environment (top right panel in Rstudio).

I am going to organise my survey battery for basic values into a new matrix (x) using the cbind function. Once you have created this, you can use the summary() function to check the descriptive statistics for each item. You can also use the cor() function check the inter-item correlations before you conduct EFA. Your correlations should be higher between items that you think tap the same latent constructs. If you have any missing values in your dataset, remember to remove these beforehand or use the na.omit() function to tell R that they should be ignored. At this stage you may also like to rename your individual measurement variables. In this example, I have relabelled each one according to the lower order value it is supposed to measure (Conformity, Tradition, Security, Benevolence, Universalism, Self- Direction, Stimulation, Hedonism, Achievement, Power).

```
x <- cbind(MP$ImpObe, MP$ImpRel, MP$ImpOrg, MP$ImpTra, MP$ImpBeh, MP$ImpSta,
MP$ImpCar, MP$ImpEqu, MP$ImpSup, MP$ImpPea, MP$ImpAhe, MP$ImpLea, MP$ImpSuc,
MP$ImpCha, MP$ImpCur, MP$ImpAdv, MP$ImpFun, MP$ImpOri, MP$ImpNew, MP$ImpEnj)
colnames(x)<- c("CONF1", "TRAD1", "SEC1", "TRAD2", "CONF2", "SEC2", "BEN1", "
UNI1", "BEN2", "UNI2", "ACH1", "POW1", "ACH2", "POW2", "SDI1", "STM1", "HED1"
, "SDI2", "STM2", "HED2")
summary(x)
cor(na.omit(x))
```

Once your data is ready to analyse, you have two more decisions to make. In the first instance, you need to choose how many factors you'd like the computer to extract from the data. You might have an a priori idea of how many latent constructs there should be in the data and you'd like to test whether this works. Alternatively, you might conduct a principal component analysis (see guidance sheet 2 in this series) and retain only the number of factors that have eigenvalues above 1. This is generally recommended. The second decision you need to make is whether or not you will include a rotation in your EFA and if so, which type. Factor rotation alters the pattern of the factor loadings, and hence can improve interpretation. Rotation can best be explained by imagining factors as axes in a graph, on which the original variables load. By rotating these axes, then, it is possible to make clusters of variables load optimally. There are three types of orthogonal rotation:

A. QUARTIMAX: Rows are simplified so that the variable should be loaded on a single factor.

B. VARIMAX: Used to simplify the column of the factor matrix so that the factor extracts are clearly associated and there should be some separation among the variables.

C. EQUIMAX: The combination of the above two methods. This method simplifies row and column at a single time.

Given that I am using measurement variables to tap basic values that are supposed to be interdependent, I am going to use varimax rotation in this example.

To run your EFA, you need to create a new object in R using the factanal() function. You can then 'run' your new object to obtain factor scores and loadings. In this example, I am going to restrict the number of factors for extraction to 4. In the theory of Basuc Human Values devised by Shalom Schwartz (1992), each of the ten lower order values (measured here by two questionnaire items each) can be clustered into four higher order values (Self- Enhancement (containing Achievement and Power values); Self-Transcendence (containing Benevolence and Universalism values); Conservation (containing Security, Tradition and Conformity values); and Openness (containing Self-Direction, Stimulation and Hedonims values). Theoretically, therefore, 4 factors should emerge from my data. Let's see.

```
fac1 <- factanal(na.omit(x), factors = 4, rotation = "varimax")
fac1

##
## Call:
## factanal(x = na.omit(x), factors = 4, rotation = "varimax")
##
## Uniquenesses:
## CONF1 TRAD1  SEC1 TRAD2 CONF2  SEC2  BEN1  UNI1  BEN2  UNI2  ACH1  POW1
## 0.594 0.658 0.870 0.257 0.699 0.513 0.379 0.714 0.459 0.766 0.272 0.423
##  ACH2  POW2  SDI1  STM1  HED1  SDI2  STM2  HED2
## 0.201 0.454 0.570 0.419 0.523 0.644 0.444 0.433
##
## Loadings:
##       Factor1 Factor2 Factor3 Factor4
## CONF1                  0.622   0.128
## TRAD1         -0.146   0.563
## SEC1           0.342
## TRAD2                  0.854  -0.102
## CONF2          0.221   0.479   0.147
## SEC2  -0.111   0.303   0.612
## BEN1   0.168                   0.763
## UNI1   0.353          -0.193   0.348
## BEN2   0.109                   0.722
## UNI2   0.226                   0.422
## ACH1   0.333   0.773           0.103
## POW1   0.411   0.612  -0.119   0.137
## ACH2   0.167   0.860   0.117   0.130
## POW2   0.145   0.723
## SDI1   0.620          -0.106   0.162
## STM1   0.752
## HED1   0.592   0.346
## SDI2   0.541          -0.210   0.138
## STM2   0.674   0.199           0.228
## HED2   0.663   0.320           0.145
##
##                Factor1 Factor2 Factor3 Factor4
## SS loadings      3.045   2.807   2.212   1.643
## Proportion Var   0.152   0.140   0.111   0.082
## Cumulative Var   0.152   0.293   0.403   0.485
##
## Test of the hypothesis that 4 factors are sufficient.
## The chi square statistic is 194.91 on 116 degrees of freedom.
## The p-value is 6.29e-06
```

Interpreting the Results

The first section of the output provides figures for the "uniqueness" of our measurement variables. We should be wary of particularly high uniqueness figures (>.7), which normally show that the variable does not fit neatly into the factors we have extracted. In this example, we can see that the first item measuring Security values has a high uniqueness of .87. We can look at this another way. If the uniqueness of a variable is substracted from 1, then we get the communality value for that variable. Communality is the variance of the nth variable explained by the factors in the model (m). In this example, my first Security item has a communality of 0.13, which means that only 13% of the variance in that variable was contributed by the 4 common factors extracted in the EFA. This result might be effected by the relatively small sample size in this example, or it might reflect that the questionnaire item is a poor measurement of Security values, or it might be that it is measuring something entirely different, or it might be the theory in question does not work as expected in the sample population. You should carefully consider your theoretical approach to the study and survey design before reaching a conclusion.

The next section provides factor loadings, which run from -1 to 1. These results are simply the correlations of each measurement variable with the unobserved factors. Rather than focusing on individual results, we are interested in groups of relatively high factor loadings (>.5). Such groupings help us to make sense of our factors. For example, we can see that the four variables measuring Achievement and Power values load strongly on the second factor, suggesting that this is the Self-Enhancement grouping we were expecting. If you scan the other results, you will see that the measurement variables load in the groupings mentioned earlier in this document. The only exceptions to this are the weaker item for Security values, and the two items measuring Universalism values. Such discrepancies are worth exploring in your write up.

The next section summarises the factors. We're particularly interested in the final row: "Cumulative Var". This provides a figure for the cumulative proportion of variance explained by the factors (ranging from 0 to 1). Ideally we want to see a high final figure, although this is generally a subjective decision. Given that the final cumulative variance accounted for by the 4 factors in this model is only .49, it looks like we should consider more than four factors. The row above, "Proportion Var", is simply the proportion of variance explained by each factor. The "SS loadings" row is the sum of squared loadings. This is sometimes used to determine the value of a particular factor. We say a factor is worth keeping if the SS loading is greater than 1. In this example, all are greater than 1.

The final section presents the results of a hypothesis test. The null of this test is that 4 factors are sufficient for our model. The low p-value leads us to reject the hypothesis and consider adding more factors. This hypothesis test is provided based on our method of estimation, maximum likelihood.

Section 2

Confirmatory Factor Analysis

What is it and when to use it?

In EFA, all of the variables load onto all factors and we use a rotation method to determine the most parsimonious and effective structure in the data (i.e. clear factors with the fewest cross-loadings). However, in confirmatory factor analysis (CFA) we must use our theoretical approach to the data to specify the number of factors and their inter-correlations, and which measurement variables load onto which factors. If EFA is relatively inductive in its approach to creating structures in your data, then CFA is far more deductive and allows you to test the theoretical assumptions you bring to the data.

In this example, I know that [theoretically] my individual measurement variables should measure ten latent lower order values and, correspondingly, four latent higher order values (Self-Enhancement (containing Achievement and Power values); Self-Transcendence (containing Benevolence and Universalism values); Conservation (containing Security, Tradition and Conformity values); and Openness (containing Self-Direction, Stimulation and Hedonims values)). Theoretically, therefore, I expect my items to load onto the respective latent variables outlined above. I will use CFA to test these assumptions.

Example

In order to run a CFA in R, you will first need to install and load the 'Lavaan' package. You can then start by specifying the model you'd like to run. In other words, you need to tell R

which measurement variables are supposed to underlie which latent factors in your model. In this example, I have grouped my measurement variables according to which of the 4 higher order values they are supposed to tap.

To estimate the model in lavaan, the easiest method is to use the 'cfa' function. It comes with sensible defaults for estimating CFA models, including the assumption that you'll want to estimate covariances among all of your latent factors (so we don't actually have to write those covariances into the model). You can get all of the information you need about your CFA model with the summary() command (see below).

The default estimator for CFA models with continuous indicators is maximum likelihood. The default treatment of missing data is listwise deletion but you can tell R to use Full Information Maximum Likelihood (FIML). Latent factors aren't measured, so they don't naturally have any scale. In order to come up with a unique solution, though, the estimator needs to have some scale for them. Lavaan's default is to set each latent factor's scale to the scale of its first indicator. Alternatively you can constrain the latent factors to have a mean of 0 and a variance of 1 (i.e. to standardize them) using the 'std.lv=TRUE' argument when you use the function 'cfa()'.

```
MODEL <-
'CON =~ ImpObe + ImpRel + ImpOrg + ImpTra + ImpBeh + ImpSta
STR =~ ImpCar + ImpEqu + ImpSup + ImpPea
SEHN =~ ImpAhe + ImpLea + ImpSuc + ImpCha
OPEN =~ ImpCur + ImpAdv + ImpFun + ImpOri + ImpNew + ImpEnj'

Fit <- cfa(MODEL, data = MP, missing = "fiml")

summary(Fit, fit.measures = TRUE, standardized = TRUE)

## lavaan 0.6-3 ended normally after 80 iterations
##
##   Optimization method                           NLMINB
##   Number of free parameters                         66
##
##                                                   Used       Total
##   Number of observations                           106         107
##   Number of missing patterns                         2
##
##   Estimator                                         ML
##   Model Fit Test Statistic                     321.639
##   Degrees of freedom                               164
##   P-value (Chi-square)                           0.000
##
## Model test baseline model:
##
##   Minimum Function Test Statistic              986.683
##   Degrees of freedom                               190
##   P-value                                        0.000
```

```
##
## User model versus baseline model:
##
##   Comparative Fit Index (CFI)                    0.802
##   Tucker-Lewis Index (TLI)                       0.771
##
## Loglikelihood and Information Criteria:
##
##   Loglikelihood user model (H0)              -3057.806
##   Loglikelihood unrestricted model (H1)      -2896.987
##
##   Number of free parameters                         66
##   Akaike (AIC)                                6247.612
##   Bayesian (BIC)                              6423.399
##   Sample-size adjusted Bayesian (BIC)         6214.881
##
## Root Mean Square Error of Approximation:
##
##   RMSEA                                          0.095
##   90 Percent Confidence Interval          0.080  0.111
##   P-value RMSEA <= 0.05                          0.000
##
## Standardized Root Mean Square Residual:
##
##   SRMR                                           0.095
##
## Parameter Estimates:
##
##   Information                                 Observed
##   Observed information based on                Hessian
##   Standard Errors                             Standard
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   CON =~
##     ImpObe            1.000                               0.817    0.590
##     ImpRel            1.118    0.265    4.227    0.000    0.913    0.501
##     ImpOrg            0.384    0.212    1.814    0.070    0.314    0.220
##     ImpTra            1.391    0.255    5.465    0.000    1.136    0.787
##     ImpBeh            0.897    0.216    4.156    0.000    0.733    0.558
##     ImpSta            1.194    0.257    4.648    0.000    0.975    0.718
##   STR =~
##     ImpCar            1.000                               0.535    0.734
##     ImpEqu            0.883    0.270    3.270    0.001    0.473    0.460
##     ImpSup            1.113    0.188    5.927    0.000    0.596    0.719
##     ImpPea            1.163    0.326    3.574    0.000    0.623    0.493
##   SEHN =~
##     ImpAhe            1.000                               1.249    0.863
##     ImpLea            0.684    0.088    7.757    0.000    0.854    0.709
##     ImpSuc            0.965    0.086   11.222    0.000    1.205    0.875
```

```
##     ImpCha            0.689    0.089    7.739    0.000    0.860    0.716
##   OPEN =~
##     ImpCur            1.000                              0.491    0.591
##     ImpAdv            1.546    0.277    5.583    0.000    0.759    0.698
##     ImpFun            1.609    0.302    5.328    0.000    0.790    0.725
##     ImpOri            1.227    0.295    4.164    0.000    0.602    0.478
##     ImpNew            1.985    0.355    5.588    0.000    0.974    0.719
##     ImpEnj            1.977    0.348    5.677    0.000    0.970    0.795
##
## Covariances:
##                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   CON ~~
##     STR              -0.001    0.057   -0.023    0.981   -0.003   -0.003
##     SEHN              0.235    0.128    1.837    0.066    0.231    0.231
##     OPEN              0.011    0.048    0.241    0.809    0.029    0.029
##   STR ~~
##     SEHN              0.156    0.083    1.885    0.059    0.233    0.233
##     OPEN              0.122    0.040    3.022    0.003    0.466    0.466
##   SEHN ~~
##     OPEN              0.369    0.095    3.880    0.000    0.603    0.603
##
## Intercepts:
##                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .ImpObe            3.953    0.134   29.425    0.000    3.953    2.858
##    .ImpRel            2.943    0.177   16.638    0.000    2.943    1.616
##    .ImpOrg            4.160    0.139   29.987    0.000    4.160    2.913
##    .ImpTra            3.337    0.141   23.742    0.000    3.337    2.313
##    .ImpBeh            4.250    0.128   33.177    0.000    4.250    3.235
##    .ImpSta            3.670    0.132   27.825    0.000    3.670    2.703
##    .ImpCar            5.302    0.071   74.874    0.000    5.302    7.272
##    .ImpEqu            5.292    0.100   53.025    0.000    5.292    5.150
##    .ImpSup            5.111    0.081   63.333    0.000    5.111    6.172
##    .ImpPea            4.606    0.123   37.408    0.000    4.606    3.648
##    .ImpAhe            3.651    0.141   25.969    0.000    3.651    2.522
##    .ImpLea            4.019    0.117   34.332    0.000    4.019    3.335
##    .ImpSuc            3.548    0.134   26.465    0.000    3.548    2.575
##    .ImpCha            3.210    0.117   27.449    0.000    3.210    2.674
##    .ImpCur            5.274    0.081   65.397    0.000    5.274    6.352
##    .ImpAdv            3.679    0.106   34.870    0.000    3.679    3.387
##    .ImpFun            3.406    0.106   32.217    0.000    3.406    3.129
##    .ImpOri            4.142    0.123   33.713    0.000    4.142    3.288
##    .ImpNew            4.194    0.132   31.769    0.000    4.194    3.095
##    .ImpEnj            3.775    0.119   31.773    0.000    3.775    3.094
##     CON               0.000                              0.000    0.000
##     STR               0.000                              0.000    0.000
##     SEHN              0.000                              0.000    0.000
##     OPEN              0.000                              0.000    0.000
##
## Variances:
##                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
```

```
##     .ImpObe          1.246    0.206    6.064    0.000    1.246    0.651
##     .ImpRel          2.484    0.386    6.428    0.000    2.484    0.749
##     .ImpOrg          1.942    0.272    7.137    0.000    1.942    0.952
##     .ImpTra          0.792    0.202    3.921    0.000    0.792    0.380
##     .ImpBeh          1.190    0.192    6.197    0.000    1.190    0.689
##     .ImpSta          0.893    0.188    4.747    0.000    0.893    0.484
##     .ImpCar          0.245    0.060    4.064    0.000    0.245    0.461
##     .ImpEqu          0.832    0.135    6.166    0.000    0.832    0.788
##     .ImpSup          0.331    0.077    4.309    0.000    0.331    0.483
##     .ImpPea          1.206    0.198    6.098    0.000    1.206    0.757
##     .ImpAhe          0.536    0.117    4.581    0.000    0.536    0.256
##     .ImpLea          0.723    0.119    6.061    0.000    0.723    0.498
##     .ImpSuc          0.445    0.103    4.314    0.000    0.445    0.234
##     .ImpCha          0.702    0.116    6.078    0.000    0.702    0.487
##     .ImpCur          0.449    0.068    6.550    0.000    0.449    0.651
##     .ImpAdv          0.604    0.100    6.053    0.000    0.604    0.512
##     .ImpFun          0.561    0.098    5.709    0.000    0.561    0.474
##     .ImpOri          1.225    0.180    6.806    0.000    1.225    0.772
##     .ImpNew          0.888    0.153    5.820    0.000    0.888    0.483
##     .ImpEnj          0.548    0.110    4.958    0.000    0.548    0.368
##      CON             0.667    0.230    2.905    0.004    1.000    1.000
##      STR             0.286    0.082    3.499    0.000    1.000    1.000
##      SEHN            1.559    0.293    5.326    0.000    1.000    1.000
##      OPEN            0.241    0.078    3.092    0.002    1.000    1.000
```

Interpreting the Results

The first few blocks of output relate to different fit indices. Rose Hartman describes each of these succinctly on her blog 'Understanding Data'. Here are some of her key definitions:

CFI (Comparative fit index): Measures whether the model fits the data better than a more restricted baseline model. Higher is better, with okay fit > .9.

TLI (Tucker-Lewis index): Similar to CFI, but it penalizes overly complex models (making it more conservative than CFI). Measures whether the model fits the data better than a more restricted baseline model. Higher is better, with okay fit > .9.

AIC (Akaike's information criterion): Attempts to select models that are the most parsimonious/efficient representations of the observed data. Lower is better. AIC is a good comparison statistic when measuring the comparative fit of different CFA models.

BIC (Schwarz's Bayesian information criterion): Similar to AIC but a little more conservative, also attempts to select models that are the most parsimonious/efficient representations of the observed data. Lower is better.

RMSEA (Root mean square error of approximation): The "error of approximation" refers to residuals. Instead of comparing to a baseline model, it measures how closely the model reproduces data patterns (i.e. the covariances among indicators). Lower is better. It comes with a 90% confidence interval in lavaan and other major SEM software, so that's often reported along with it.

WARNING: It is very important to remember that decent 'fit' statistics do not automatically mean that you have produced a good model, or that the model is a truthful representation of relationships in real life. You can often alter your model in any number of ways and still extract good fit statistics. It is only in the context of your chosen theory that you can decide whether or not a model fits well.

After the fit indices, R provides you with a series of parameter estimates. These results show you the factor loadings of each measurement variable. For example, in this dataset the item ImpRel (which measures Tradition values) has a factor loading on the factor Conservation of 1.18. Factor loadings can be interpreted like a regression coefficient. For example, this parameter tells us that for each unit increase in the latent factor Conservation (since we standardized latent factors, this means for each 1SD increase), the model predicts a 1.18-unit increase in ImpRel.

The output is rather long and lots of it might not be useful and/or worth reporting. For the purpose of interpreting your results and making them presentable, you can pull out all of the factor loadings and put them in their own table with functions from the 'dplyr' package, and 'kable' from the 'knitr' package.

```
parameterEstimates(Fit, standardized=TRUE) %>%
  filter(op == "=~") %>%
  select('Latent Factor'=lhs, Indicator=rhs, B=est, SE=se, Z=z, 'p-value'=pva
lue, Beta=std.all) %>%
  kable(digits = 3, format="pandoc", caption="Basic Values: Factor Loadings")
```

Basic Values: Factor Loadings

Latent Factor	Indicator	B	SE	Z	p-value	Beta
CON	ImpObe	1.000	0.000	NA	NA	0.590
CON	ImpRel	1.118	0.265	4.227	0.000	0.501
CON	ImpOrg	0.384	0.212	1.814	0.070	0.220
CON	ImpTra	1.391	0.255	5.465	0.000	0.787
CON	ImpBeh	0.897	0.216	4.156	0.000	0.558
CON	ImpSta	1.194	0.257	4.648	0.000	0.718
STR	ImpCar	1.000	0.000	NA	NA	0.734
STR	ImpEqu	0.883	0.270	3.270	0.001	0.460
STR	ImpSup	1.113	0.188	5.927	0.000	0.719
STR	ImpPea	1.163	0.326	3.574	0.000	0.493
SEHN	ImpAhe	1.000	0.000	NA	NA	0.863
SEHN	ImpLea	0.684	0.088	7.757	0.000	0.709
SEHN	ImpSuc	0.965	0.086	11.222	0.000	0.875
SEHN	ImpCha	0.689	0.089	7.739	0.000	0.716
OPEN	ImpCur	1.000	0.000	NA	NA	0.591
OPEN	ImpAdv	1.546	0.277	5.583	0.000	0.698
OPEN	ImpFun	1.609	0.302	5.328	0.000	0.725
OPEN	ImpOri	1.227	0.295	4.164	0.000	0.478
OPEN	ImpNew	1.985	0.355	5.588	0.000	0.719
OPEN	ImpEnj	1.977	0.348	5.677	0.000	0.795

It is possible that you may have competing theories about how your variables should interact and what they should be tapping (different unobservable factors). In that case, you can run a series of different CFA models with different specifications and compare the fit statistics of each model against the others.

When it comes to writing up your results, Alexander Beaujean (2014) recommends ten essential components:

1. A theoretical and empirical justification for the hypothesized model.

2. A complete description of how the model was specified (i.e., the indicator variables for each latent variable, the scaling of the latent variables, a description of what parameters were estimated and constrained).

3. A description of your sample (i.e., demographic information, sample size, sampling method).

4. A description of the type of data used (e.g., nominal, continuous) and descriptive statistics.

5. Tests of assumptions (specifically that the indicator variables follow a multivariate normal distribution) and the type of estimator used in your model.

6. A description of missing data and how the missing data was handled.

7. The software and version used to fit the model.

8. Measures, and the criteria used, to judge model fit.

9. Any alterations made to the original model based on model fit or modification indices.

10. All parameter estimates (i.e., loadings, error variances, latent (co)variances) and their standard errors.

References

Beaujean, A. (2014). Factor Analysis Using R. Practical Assessment, Research & Evaluation, Vol 18, No 4, pp.1-11

Rietveld, T. & Van Hout, R. (1993). Statistical Techniques for the Study of Language and Language Behaviour. Berlin - New York: Mouton de Gruyter.

Schwartz, S. (1992). Universals in the content and structure of values: Theory and empirical tests in 20 countries. In M. Zanna (ed.). Advances in experimental social psychology. New York: Academic Press.