# R Instructions

The following exercise builds on the previous exploratory factor analysis exercise. In the data folder you have exploratory and confirmatory SPSS data files. We'll start with the exploratory data file. For final reporting purposes you might want to run all the retained models using the confirmatory data file.

The aim of the following analyses is to assess the relative model fit of various models of this personality test and to develop a good model.

We will fit the models using all 25 items, but note that potentially you could exclude one or more of the items from the set of analyses. Nonetheless, to make model comparisons meaningful it is generally best to use a common set of items. Thus, if you delete a exclude a few items, exclude them from all models.

There are many possible models that you could fit.

1. We will fit:

(a) a one factor model

(b) a five factor model with uncorrelated factors

(c) a five factor model with all factors inter-correlated

(d) a more data driven model where we take one of the existing models and add some reasonable modifications to improve the fit (add a justification based on theory and modification indices for this model)

2. Create a table with the model fits for each model (i.e., rows are models, columns are fit statistics): include at least chi-square, df, RMSEA, SRMR, and CFI.

# Answers

(see also the completed example project)

## Getting Started

Download and unzip ProjectTemplate folder structure

<https://github.com/jeromyanglim/AnglimModifiedProjectTemplate/archive/master.zip>

Move the top level folder to a suitable location on your hard drive

Rename the parent folder and the RStudio-Project file to be a clear name.

e.g., "confirmatory-factor-analysis-exercise.Rproj"

Double click on the RStudio project file to open Project in RStudio

This ensures that you have the correct working directory.

We first need to load the data.

So copy the data file (e.g., "bfi-unique-3-exploratory.sav") to the "data" directory of your project. You should then rename it so that it corresponds to the name you want it to have in R. I often use the label "rcases" or "ccases" to correspond to "raw participants" and "cleaned participants", but you can call it any legal R name. Generally 3 to 7 characters is good for a data frame name.

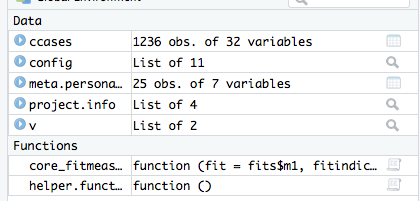
You may also want to be able to load the meta-data for the personality test (i.e., "bfi-personality-meta-data.xlsx".

So put that in the "data" folder as well, and rename it to something like "meta.xlsx". In general, when you import Excel file using ProjectTemplate, the data.frame will get the name: filename.sheetname.

You can then load the data and the metadata by running the command:

library(ProjectTemplate); load.project()

This code is shown in the "explore.rmd" file.



Basic data preparations:

I put the following code in the munge folder.

names(ccases) <- tolower(names(ccases))

v <- list()

v$items <- meta.personality$name

v$fitindicies <- c("npar", "chisq", "df", "pvalue", "cfi", "rmsea",

"rmsea.ci.lower", "rmsea.ci.upper", "srmr")

* It makes the variable names lower case which means that you then don't have to think about case when working with the variables
* I also create a list of various things.
* In particular, v$fitindicies represents the names of the key fit indices that we'll use later.

## Quick check on an EFA

fac1 <- factanal(ccases[ , v$items], factors = 5,

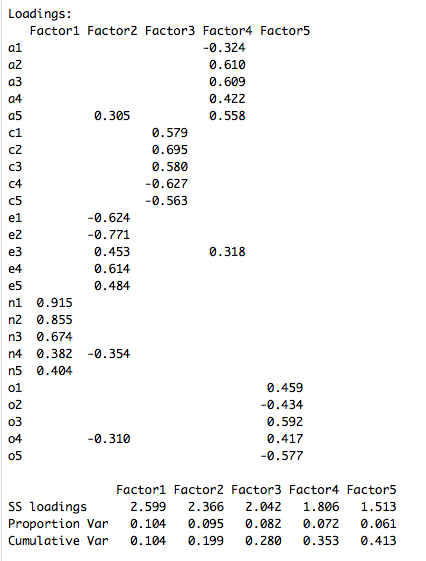
rotation = "promax")

psych::scree(ccases[ , v$items])

print(fac1, cutoff = .3)

?loadings

The above code runs a basic exploratory factor analysis extracting five factors and using a promax rotation.



* This shows that the rotated factor loadings with values below .30 suppressed.
* Most items are loading consistently with other items from their respective factors.
* a5, e3, n4, and o4 are all absolute cross-loadings above .30. They're not huge, but they're probably worth noting as they may be relevant to the CFA stage if you want to improve the model.

## Getting started with lavaan

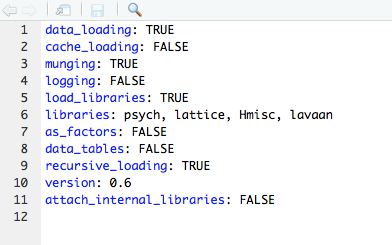
If you want to use lavaan, you need to have installed

install.packages("lavaan")

If you already have it installed, you still need to load it. This usually involves running:

library(lavaan)

But if you are using ProjectTemplate, then you add "lavaan" to your list of libraries in "config/global.dcf". Here's an example; yours might look slightly different, but the point is that "lavaan" is listed in the libraries section.



## (a) a one factor model

Before getting started with fitting specific models, I like to create a couple of list objects to store the respective models and fit objects.

models <- list()

fits <- list()

* These are just empty lists for now, but we'll progressively add things to them.
* Storing the models and fits in lists allows us to loop over them at a later stage to get fit statistics and so on for all the fits.

The one factor model can be specified as follows

models$m1 <-

'global =~ a3 + a1 + a2 + a4 + a5 + c1 + c2 + c3 + c4 +

c5 + e1 + e2 + e3 + e4 + e5 + n1 + n2 + n3 +

n4 + n5 + o1 + o2 + o3 + o4 + o5'

* "global" is the name we have given the latent variable representing the global factor
* "=~" is the operator that separate the latent variable from the indicators.
* "a3, a1, ... o5" are the variable names corresponding to the 25 personality items
* Each item is separated by a plus
* The whole string is placed in quotation marks. In this sense, it is stored as a regular string variable in R, which in this case I have assigned to the variables "models$m1".
* I made a3 the first item, because this automatically has the unstandardized loading of 1 which determines the orientation of this global factor, and a3 is a socially desirable item whereas a1 is not.

To run this model, we do the following:

fits$m1 <- lavaan::cfa(models$m1, data = ccases)

* cfa is the name of the function; the "lavaan::" bit is optional. It just makes it explicit that I am referring to the "cfa" function in the lavaan package. It just so happens that there is another package that also has a cfa function. That said, it's unlikely that you'll also have loaded.
* The first argument takes the model specification (i.e., models$m1)
* We then supply the data
* The resulting fit object can then be assigned to a variable.

**Interpret Output:** There are a various functions for extracting output.

* I have this cheet sheet: http://jeromyanglim.tumblr.com/post/33556941601/lavaan-cheat-sheet
* You can also check out the various tutorials on the lavaan website: http://lavaan.ugent.be/tutorial/index.html
* Or you can browse the lavaan help files.

To get an overview of the model, you can use

summary(fits$m1, fit.measures = TRUE)

* summary is a generic method that uses the class of the first argument to determine what function to call. Because fits$m1 is a cfa fit, then it calls the summary function appropriate for lavaan cfa fits.

The output includes basic model information, fit statistics, and parameter estimates:

lavaan (0.5-23.1097) converged normally after 39 iterations

Number of observations 1236

Estimator ML

Minimum Function Test Statistic 5460.242

Degrees of freedom 275

P-value (Chi-square) 0.000

* "Converged" means that the model appears to have achieved a stable solution
* 1236 is the number of cases
* ML indicates that you uses maximum likelihood estimation
* Minimum function test statistics: 5460 is your chi-square: This is an overall measure of discrepancy between the model and the data. It is influenced both by the sample size and the size of the discrepancy. Higher values mean poorer fit. However, it should rarely be interpreted in isolation. Rather, it is useful when comparing different models, and it is also useful as a basis for creating other fit indices.

**Degrees of freedom:** I.e., 325 - 50=275 degrees of freedom.

Standardised regression weights:

standardizedSolution(fits$m1)

You get the following abbreviated output

lhs op rhs est.std se z pvalue

1 global =~ a3 0.517 0.023 22.117 0.000

2 global =~ a1 -0.205 0.030 -6.918 0.000

3 global =~ a2 0.463 0.025 18.622 0.000

4 global =~ a4 0.394 0.026 14.869 0.000

5 global =~ a5 0.642 0.020 32.924 0.000

6 global =~ c1 0.305 0.028 10.788 0.000

7 global =~ c2 0.284 0.029 9.928 0.000

8 global =~ c3 0.289 0.028 10.139 0.000

9 global =~ c4 -0.349 0.027 -12.742 0.000

10 global =~ c5 -0.431 0.026 -16.812 0.000

11 global =~ e1 -0.479 0.024 -19.592 0.000

12 global =~ e2 -0.639 0.020 -32.574 0.000

13 global =~ e3 0.566 0.022 25.792 0.000

14 global =~ e4 0.629 0.020 31.491 0.000

15 global =~ e5 0.530 0.023 22.981 0.000

16 global =~ n1 -0.337 0.028 -12.192 0.000

17 global =~ n2 -0.348 0.027 -12.684 0.000

18 global =~ n3 -0.331 0.028 -11.905 0.000

19 global =~ n4 -0.449 0.025 -17.781 0.000

20 global =~ n5 -0.261 0.029 -9.012 0.000

21 global =~ o1 0.312 0.028 11.103 0.000

22 global =~ o2 -0.135 0.030 -4.450 0.000

23 global =~ o3 0.398 0.026 15.071 0.000

24 global =~ o4 -0.026 0.031 -0.850 0.395

25 global =~ o5 -0.153 0.030 -5.084 0.000....

26 a3 ~~ a3 0.732 0.024 30.239 0.000

27 a1 ~~ a1 0.958 0.012 78.822 0.000

28 a2 ~~ a2 0.786 0.023 34.127 0.000

...

48 o3 ~~ o3 0.842 0.021 40.043 0.000

49 o4 ~~ o4 0.999 0.002 619.675 0.000

50 o5 ~~ o5 0.976 0.009 105.615 0.000

51 global ~~ global 1.000 0.000 NA NA

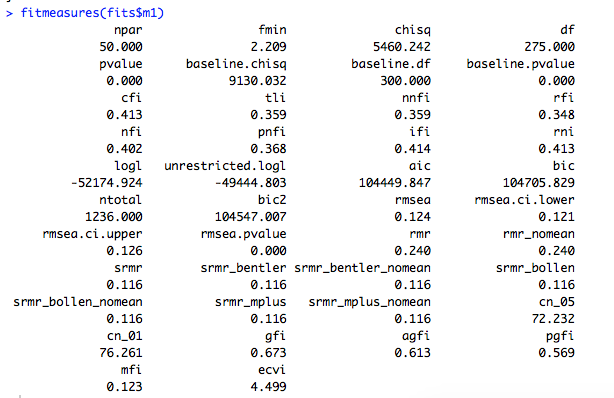
* Standardized loadings on latent variables all have the operator ("op") "=~". i.e., "lhs" is left-hand side; rhs is right-hand side. "est.std" is standardized estimate. se is the standard error and z and p are related to whether the coefficient is signifciantly different from zero.
* Thus, if you look at this, you can see which standardized loadings are smaller and larger. For example, e2 (-), e4 (+), and a5 (+) have large coefficients. These concern difficulties with others (e2), making friends easily (e4) and loving children. From past experience, I find that the first factor tends to be related to social desirability and social effectiveness. Perhaps these are good examples fo this.
* The residuals are denoted by "~~" (i.e., variance) and having the same item name on lhs and rhs. So for example, a3 has a standardised residual variance of 0.732. I.e., 73% of the variance in a3 is not explained by the global factor.

Overall, it shows that certain items load strongly on the global factor such as E2, E4, A3, A5, etc. and some items load weakly (e.g., O4, O2, etc.). By looking at the direction of the loadings and the size of the loadings we can see that high scores seem to reflect more negative emotions, fewer positive emotions, and perhaps lower self-esteem. In general, high scores seem to relate to a social undesirability factor, whether this be real or the result of a response bias.

Regardless, of the specific issues here, you should observe how the meaning of latent variables can be understood in the same way as you would interpret a factor analysis factor. You should also note that regardless of model fit, it is still important to try to understand the meaning of the model in terms of what the coefficients mean.

**Model fit**

lavaan provides a wide range of model fit information.



I'll focus on chi-square, df, RMSEA, SRMR, and CFI. In general it is a good idea to have a separate table where you record this information for each of your models.

If you just want to extract a few fit measures, you can extract them by name. Here is the complete list extracted using dput:

> fm <- fitmeasures(fits$m1)

> dput(names(fm))

c("npar", "fmin", "chisq", "df", "pvalue", "baseline.chisq",

"baseline.df", "baseline.pvalue", "cfi", "tli", "nnfi", "rfi",

"nfi", "pnfi", "ifi", "rni", "logl", "unrestricted.logl", "aic",

"bic", "ntotal", "bic2", "rmsea", "rmsea.ci.lower", "rmsea.ci.upper",

"rmsea.pvalue", "rmr", "rmr\_nomean", "srmr", "srmr\_bentler",

"srmr\_bentler\_nomean", "srmr\_bollen", "srmr\_bollen\_nomean", "srmr\_mplus",

"srmr\_mplus\_nomean", "cn\_05", "cn\_01", "gfi", "agfi", "pgfi",

"mfi", "ecvi")

So for example, I could run:

> fm[c("chisq", "df", "rmsea", "srmr", "cfi")]

chisq df rmsea srmr cfi

5460.2421856 275.0000000 0.1235119 0.1162352 0.4127720

>

* chi-square value for the model where larger values mean greater discrepancy between model and data
* DF is the number of degrees of freedom
* RMSEA is a measure of model discrepancy. A rule of thumb is that RMSEA < .05 represents close fit and <.08 represents reasonable fit. Clearly, here we have poor fit.
* Standardized RMR: A rough idea of SRMR can be to think of it as the typical absolute difference between model and data correlations, although this is a simplification. The point is that values closer to zero are good. Some have suggested ( http://davidakenny.net/cm/fit.htm ) that values below .08 are good.
* CFI is a commonly reported fit measure. It ranges from 0 to 1. Values above .9 or .95 are often considered indicators of good model fit. But once again, it is often best to interpret it in a comparative way. Nonetheless, we weren't expecting a one factor model to be very good, and this is coming across in the CFI.

## (b) a five factor model with uncorrelated factors

A five factor model can be specified as follows:

models$m2 <-

' agree =~ a3 + a1 + a2 + a4 + a5

consc =~ c1 + c2 + c3 + c4 + c5

extra =~ e3 + e1 + e2 + e4 + e5

neuro =~ n1 + n2 + n3 + n4 + n5

open =~ o1 + o2 + o3 + o4 + o5'

* It's similar to the previous model, except that now there are five latent variables each with =~ and then the items that represent the indicator for the latent variable.
* I've updated the list variable name (i.e., "m2")
* And note that e3 was placed first because this is a positively worded item (know how to captivate people", but e1 is reversed (i.e., "don't talk a lot). If e1 was put first the factor would measure introversion.

Estimate the model

fits$m2 <- lavaan::cfa(models$m2, data = ccases, orthogonal = TRUE)

* This is the same as with model 1, but notice the "orthogonal = TRUE" argument. By default, the cfa function add correlations between latent variables. So in order to disable this or specify such correlations manually, you need to specify "orthogonal = TRUE".
* You can learn more about arguments to cfa by going to ?cfa. Although somewhat unhelpfully, this argument is listed under ?lavOptions

I'll save looking at the fit measures until the end, but we can have a quick look at the standardized parameter estimates.

standardizedSolution(fits$m2)

lhs op rhs est.std se z pvalue

1 agree =~ a3 0.723 0.022 33.096 0

2 agree =~ a1 -0.340 0.030 -11.397 0

3 agree =~ a2 0.655 0.023 28.472 0

4 agree =~ a4 0.465 0.027 17.023 0

5 agree =~ a5 0.653 0.023 28.355 0

6 consc =~ c1 0.585 0.025 23.523 0

7 consc =~ c2 0.646 0.023 27.483 0

8 consc =~ c3 0.555 0.026 21.713 0

9 consc =~ c4 -0.629 0.024 -26.373 0

10 consc =~ c5 -0.567 0.025 -22.424 0

11 extra =~ e3 0.571 0.024 24.190 0

12 extra =~ e1 -0.619 0.022 -27.833 0

13 extra =~ e2 -0.762 0.019 -40.801 0

14 extra =~ e4 0.680 0.021 32.983 0

15 extra =~ e5 0.518 0.025 20.714 0

16 neuro =~ n1 0.819 0.014 58.107 0

17 neuro =~ n2 0.806 0.014 55.807 0

18 neuro =~ n3 0.712 0.017 41.112 0

19 neuro =~ n4 0.531 0.023 22.971 0

20 neuro =~ n5 0.482 0.024 19.731 0

21 open =~ o1 0.505 0.031 16.517 0

22 open =~ o2 -0.418 0.032 -13.170 0

23 open =~ o3 0.643 0.030 21.515 0

24 open =~ o4 0.351 0.033 10.723 0

25 open =~ o5 -0.545 0.030 -18.045 0

* These show the standardized loadings on the latent factors. Note how they are all pretty large. Perhaps only

## (d) a five factor model with all factors inter-correlated

This model assumes that all of the big 5 factors are intercorrelated to potentially varying degrees.

This is the same model specification as the previous model, so we can just create a copy.

models$m3 <- models$m2

We just need to change the estimation code to:

fits$m3 <- lavaan::cfa(models$m3, data = ccases, orthogonal = FALSE)

Or equivalently, given that FALSE is the default.

fits$m3 <- lavaan::cfa(models$m3, data = ccases)

**Output interpretation:**

This will give you some additional output such as the correlations between latent factors:

standardizedSolution(fits$m3)

lhs op rhs est.std se z pvalue

56 agree ~~ consc 0.337 0.035 9.567 0.000

57 agree ~~ extra 0.703 0.024 29.054 0.000

58 agree ~~ neuro -0.242 0.034 -7.116 0.000

59 agree ~~ open 0.319 0.037 8.546 0.000

60 consc ~~ extra 0.337 0.034 9.757 0.000

61 consc ~~ neuro -0.250 0.034 -7.274 0.000

62 consc ~~ open 0.282 0.038 7.365 0.000

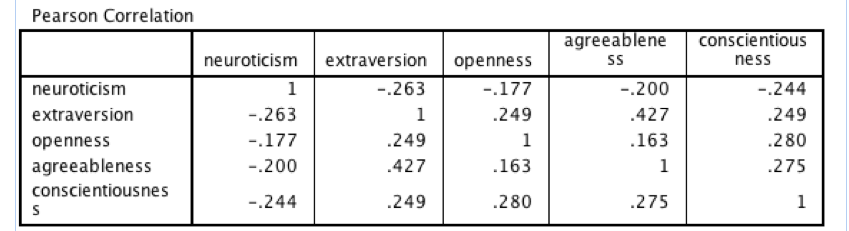
63 extra ~~ neuro -0.250 0.033 -7.510 0.000

64 extra ~~ open 0.436 0.034 12.651 0.000

65 neuro ~~ open -0.126 0.037 -3.388 0.001

>

* Note that these are the correlations between latent variables. Compare this to the correlations between scale scores from the previous exercise (albeit a few items are different):



In general, such correlations in CFA should be larger because they are in a sense adjusting for measurement error and estimating correlations between the latent variables assumed to underlie the observed items.

We can also see that certain correlations are larger than others. For example A and E correlate quite highly which may explain their high loading on the global factor.

## Combined fit table:

I wrote the following function to extract and round the fit measures of an fit object:

core\_fitmeasures <- function(fit = fits$m1,

fitindicies = c("npar", "chisq", "df", "pvalue", "cfi",

"rmsea", "rmsea.ci.lower", "rmsea.ci.upper", "srmr")

, digits = 3) {

x <- fitmeasures(fit)

round(x[fitindicies], digits)

}

* It takes a fit object and returns the specified fit measures rounded to 3 decimal places.
* If you load the function, you can then use it in your code

In particular, the following code loops over the model fits and extract the core fit measures for each and returns the results in a nice matrxi:

sapply(fits, function(X) core\_fitmeasures(X))

* sapply is a function for looping over the elements of the first argument
* Thus, on each iteration of the loop it sends one fit object to the function
* and X is replaced with that fit object
* It then returns the results in a simplified form where possible

> sapply(fits, function(X) core\_fitmeasures(X))

m1 m2 m3

npar 50.000 50.000 60.000

chisq 5460.242 3087.668 2365.090

df 275.000 275.000 265.000

pvalue 0.000 0.000 0.000

cfi 0.413 0.681 0.762

rmsea 0.124 0.091 0.080

rmsea.ci.lower 0.121 0.088 0.077

rmsea.ci.upper 0.126 0.094 0.083

srmr 0.116 0.136 0.078

So, you get the following:

So model 3 is an improvement over the previous model, but the improvement is quite small in comparison to the improvement that resulted from going from uncorrelated to either model 3 or this model.

## (e) a more data driven model where we take one of the existing models and add some reasonable modifications to improve the fit (add a justification based on theory and modification indices for this model)

There are many more models that we could explore. At this point I will use the opportunity to highlight the role of modification indices to see if it suggests any important cross-loadings or possibly correlated residuals.

Given the five factor correlated model is the best so far, I'll use that as the starting point.

To get modification indices, you can use the "modificationindices" function:

mod\_ind <- modificationindices(fits$m3)

head(mod\_ind[order(mod\_ind$mi, decreasing=TRUE), ], 10)

* The first line saves the modification indices to an R object. This object has over 275 rows, so it's helpful to pull out the modifications that are going to result in the biggest changes.
* So ordering by mi in decreasing order orders the matrix so the biggest changes are displayed at the top and head shows just the first 10 rows.

> head(mod\_ind[order(mod\_ind$mi, decreasing=TRUE), ], 10)

lhs op rhs mi epc sepc.lv sepc.all sepc.nox

421 n1 ~~ n2 172.808 0.780 0.780 0.332 0.332

95 consc =~ e5 85.501 0.564 0.392 0.300 0.300

119 extra =~ n4 79.427 -0.463 -0.394 -0.257 -0.257

156 open =~ e3 65.913 0.674 0.388 0.286 0.286

438 n3 ~~ n4 65.210 0.393 0.393 0.162 0.162

135 neuro =~ c5 62.348 0.293 0.371 0.231 0.231

124 extra =~ o4 60.792 -0.417 -0.355 -0.289 -0.289

110 extra =~ a5 60.018 0.618 0.527 0.423 0.423

276 c1 ~~ c2 56.984 0.298 0.298 0.188 0.188

123 extra =~ o3 56.879 0.484 0.412 0.333 0.333

* The M.I. is the modification index. It is a lower bound estimate of the reduction in chi-square that would result from adding that particular path.
* So it seems to be suggesting that adding a correlated residual between n1 and n2 would improve the model.

|  |  |
| --- | --- |
| N2 | Get irritated easily. |
| N3 | Have frequent mood swings. |

Perhaps these two items are particularly similar; more similar than other neuroticism items.

We can add this as an additional model and estimate it:

models$m4 <-

' agree =~ a3 + a1 + a2 + a4 + a5

consc =~ c1 + c2 + c3 + c4 + c5

extra =~ e3 + e1 + e2 + e4 + e5

neuro =~ n1 + n2 + n3 + n4 + n5

open =~ o1 + o2 + o3 + o4 + o5

# correlated residuals

n1 ~~ n2

'

fits$m4 <- lavaan::cfa(models$m4, data = ccases, orthogonal = FALSE)

If we re-run the fit measure code, we get the following:

> sapply(fits, function(X) core\_fitmeasures(X))

m1 m2 m3 m4

npar 50.000 50.000 60.000 61.000

chisq 5460.242 3087.668 2365.090 2216.181

df 275.000 275.000 265.000 264.000

pvalue 0.000 0.000 0.000 0.000

cfi 0.413 0.681 0.762 0.779

rmsea 0.124 0.091 0.080 0.077

rmsea.ci.lower 0.121 0.088 0.077 0.074

rmsea.ci.upper 0.126 0.094 0.083 0.080

srmr 0.116 0.136 0.078 0.076

We are starting to see gradual improvements and we could add more to improve fit.

In general, the modification indices approach is quite data driven. This is why you might want to split your data file into two: one for data driven exploration and another for confirmation.

If you wanted to export this fit table to a word processor, you could do the following:

sumtable <- sapply(fits, function(X) core\_fitmeasures(X))

sumtable <- t(sumtable)

sumtable

write.csv(sumtable, file = "output/sumtable.csv")

* The key bit is write.csv which will write the data.frame or matrix to a csv file
* You can then open that file in Excel and format the way you would like for your report.

## Summary thoughts

Hopefully through this process, you are getting a sense of how a model comparison approach is applied using confirmatory factor analysis.

* It is best to have several models. Start with simple naive models and gradually add complexity. Interpret fit statistics relative to other models. Justify additional complexity (i.e., fewer df) by the theory and improvement in fit.
* Models have fit statistics which guide us in deciding whether the model is an acceptable representation of the data. They also have regression weights, variances, and covariances (and their standardised versions) which help to explain what the model means.