Latent Variable Modelling Workflow Reference

Alex Rand

10/25/22

Table of contents

Preface

What is this?

This is a book full of code to use when you want to do latent variable modelling. It gives suggested workflows I've cobbled together from a few different textbooks, and has worked examples with data from those textbooks or from open datasets I found online. When you need to do latent variable modelling for your research, you can use these workflows as a place to start.

Specifically, it seems like these are the sub-areas of latent variable modelling to know how to do:

- Exploratory Factor Analysis;
- Confirmatory Factor Analysis;
- Item Response Theory;
- Full SEM;
- Longitudinal SEM.

Maybe I'll discover some other types of things along the way. It's a lifelong journey haha.

What am I referencing?

The first book on latent variable modelling I read was Gorsuch (1983). This was a nice conceptual introduction, but the applied examples were pretty whack. I've since found a few sources with data and R code to work with:

- *Latent Variable Modelling with R*, by Finch (2015). They helpfully provide all of the datasets here.
- *Principles and Practice of Structural Equation Modeling*, by Kline (2011). The publisher provides data and code here.
- The *lavaan* documentation has some nice worked examples too.

I'll mostly be using **lavaan** and **tidyverse**, but maybe also some **brms** at some point.

1 Introduction

This is a book created from markdown and executable code.

```
1 + 1
```

```
[1] 2
```

2 CFA

Let's load the packages we'll need for what is to come in this chapter:

```
library(tidyverse)
library(lavaan)
```

2.1 Example 1: Toxic Striving Energy

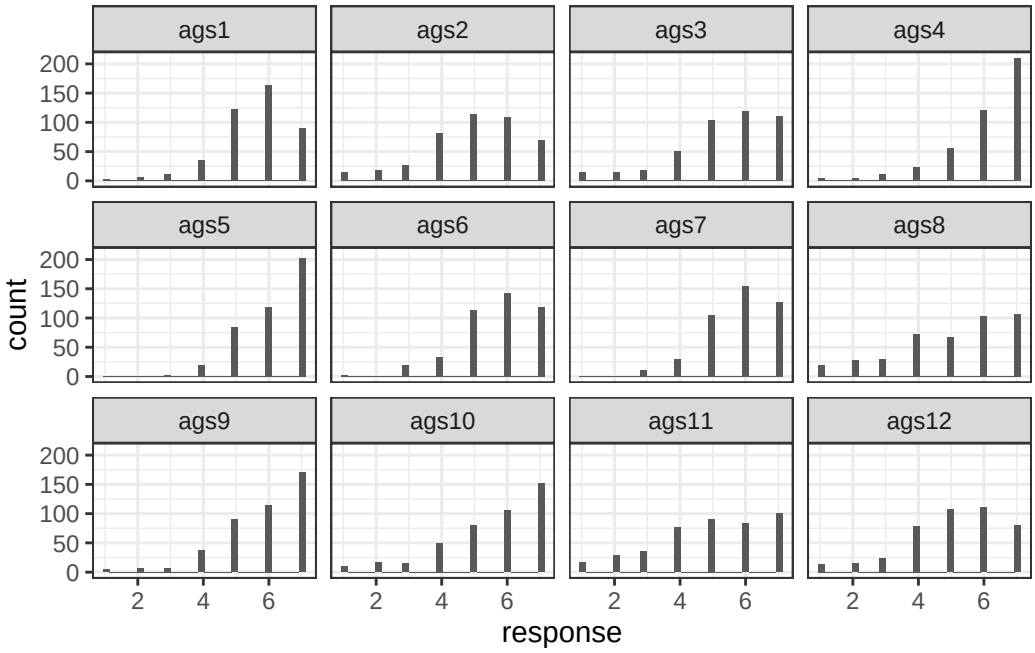
The first example we'll look at is from Finch (2015). chapter 3. The practice dataset is introduced on page 10. It is from a study about human motivation. The dataset is a weird questionnaire called the 'Achievement Goal Scale' (AGS), which asks people 12 questions about how much toxic striving energy they have. The dataset provided seems to have lots of mysterious columns in it, but we're probably good to just keep the columns with responses to the AGS questionnaire:

```
### Load the data
dat_raw <- foreign::read.spss('data/finch-and-french/edps744.sav')

### Clean the data
dat_ags <- dat_raw %>%

  # Convert to a data frame for ease of use
  as.data.frame() %>%

  # Keep only columns that start with the prefix 'ags' followed by a question number
  select(matches("ags\\d"))
```

2.1.1 Data Exploration

We don't want to do too much exploration before fitting our factor models, because the whole game of CFA is to commit to our hypotheses before checking what the data looks like, so we don't mislead ourselves with forking paths. But just for fun, we can explore the distributions of the answers to each of the 12 questions:

```r
dat_ags %>%

  # Pivot to prepare the data for visualization
  pivot_longer(
    cols      = everything(),
    names_to  = "question",
    values_to = "response",
    names_transform = list(question = fct_inorder)
  ) %>%

  # Plot
  ggplot() +
  geom_histogram(aes(x = response)) +
  theme_bw() +
  facet_wrap(~question)
```



Seems like some questions have different means and variances from each other. For example, the answers to `ags11` and `ags12` are relatively flat, while the answers to `ags4` and `ags5` are more bunched up around the highest values. The responses clearly skew towards higher values in aggregate.

We can also do some healthy exploration of missingness in the dataset. For starters: what

proportion of values are missing in each row?

```
dat_ags %>%

  # Calculate the proportion of missing values
  summarise_all(~ sum(is.na(.)) / (sum(is.na(.) + sum(!is.na(.))))) %>%

  # Rounding to make the results more presentable
  mutate(across(everything(), round, 6)) %>%

  # Create the table
  knitr::kable(title = "Proportion of Missing Responses in Each Column")
```

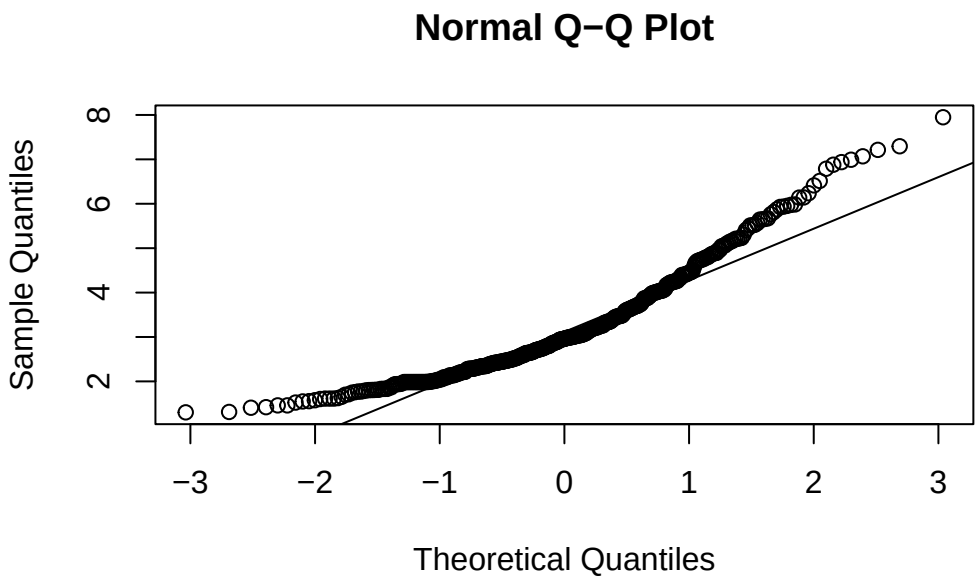
ags1	ags2	ags3	ags4	ags5	ags6	ags7	ags8	ags9	ags10	ags11	ags12
1.1e-05	5e-06	5e-06	1.6e-05	1.6e-05	1.1e-05	1.6e-05	1.6e-05	1.1e-05	2.2e-05	1.1e-05	1.6e-05

That's very little missingness. Probably no need to do multiple imputation here.

The authors also do a preliminary test of whether the responses are normally distributed, since this is one of the fundamental assumptions of maximum likelihood estimation. Kristoffer Magnusson has created a cool interactive teaching tool that nicely illustrates this point. It is worth remembering that we *do not* make this type of assumption for linear regression in general – only for maximum likelihood estimates. All we need assume for linear regression is that the *residuals* are normally distributed, as opposed to the data themselves. This common misunderstanding leads to what Richard McElreath has called 'histomancy'.

To evaluate the assumption of normalness underlying maximum likelihood estimation, the authors do what seems to be a multivariate version of a classic 'normal probability plot'. These are explained nicely in this stack exchange thread. They also produce some of the classic tests of skew and kurtosis, which I don't want to get into here. This youtuber has nice introductory videos about these topics.

```
# Run the Mardia tests for normalness
mardia.object <- psych::mardia(dat_ags)
```




```
# Plot the multivariate version of the normal probability plot
plot(mardia.object)

# Present the outputs we're interested in
tibble(
  "Skew" = mardia.object$skew,
  "Skew p-value" = mardia.object$p.skew,
  "Kurtosis" = mardia.object$kurtosis,
  "Kurtosis p-value" = mardia.object$p.kurt
) %>%

  knitr::kable()
```

Skew	Skew p-value	Kurtosis	Kurtosis p-value
2359.475	0	40.52999	0

The plotted points don't seem to fit the straight line super well, which suggests that the normalness assumption may not hold here. Also, the hypothesis tests for skew and kurtosis return some mighty low p-values, suggesting that we've got lots of each of them. So maybe maximum likelihood estimation isn't such a good idea here?

The authors proceed with it anyway for pedogogical reasons, because they want to illustrate how the maximum likelihood estimates differ from estimates arrived at using other methods.

> In actual practice, given the lack of multivariate normality that seems apparent in the previous results, we would likely not use ML and instead rely on the alternative estimation approach.

2.1.2 Model Fitting

The researchers who collected the data do what good factor analysts do: they look to the literature to set up some clear and specific candidate hypotheses, and see the degree to which this new data is compatible with each of them.

One of the candidate hypotheses is that a person's toxic striving energy ('achievement goal orientedness'?) is secretly driven by four platonic unobservable things, namely:

1. Mastery Approach 'MAP' (eg. *"I want to learn as much as possible")*;
2. Mastery Avoidant 'MAV' (eg. *"I want to avoid learning less than I possibly could"*);
3. Performance Approach 'PAP' (eg. *"I want to do well compared to other students");*
4. Performance Avoidant 'PAV' (eg. *"It is important for me to avoid doing poorly compared to other students"*)

We'll call the above hypothesis **H1.** But there's another hypothesis that says actually the 'Mastery' variables are just one monolithic thing, so really there are only 3 factors, namely 'Mastery', 'PAP', and 'PAV'. We'll call this one **H2.**

These will be the two candidate hypotheses we're gonna test via factor analysis.

The way **lavaan** works is that you need to separately define the model syntax as a string, and then feed that string to one of the model-fitting functions like `cfa()` . Then we can call the `summary()` function to get a big table of outputs.

```
# Define the relationships from my hypothesis
h1.definition <-
'map=~ags1+ags5+ags7
mav=~ags2+ags6+ags12
pap=~ags3+ags9+ags11
pav=~ags4+ags8+ags10'

# Fit the model
h1.fit <- cfa(
  data  = dat_ags,
```

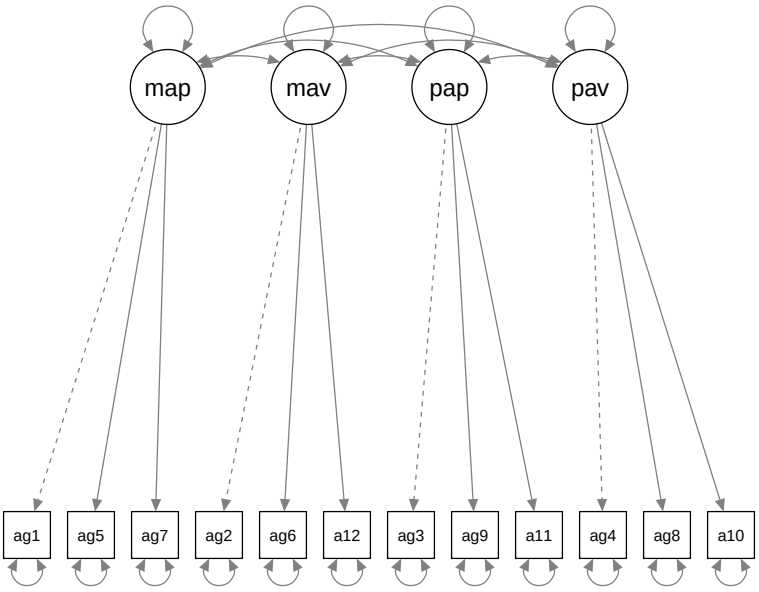
```
  model = h1.definition
)

# Look at the results
summary(h1.fit, fit.measures = TRUE)
```

```
lavaan 0.6-12 ended normally after 48 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of model parameters                        30

                                                  Used       Total
  Number of observations                           419         432

Model Test User Model:

  Test statistic                               328.312
  Degrees of freedom                            48
  P-value (Chi-square)                          0.000

Model Test Baseline Model:

  Test statistic                              3382.805
  Degrees of freedom                            66
  P-value                                       0.000

User Model versus Baseline Model:

  Comparative Fit Index (CFI)                   0.915
  Tucker-Lewis Index (TLI)                      0.884

Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)             -7014.070
  Loglikelihood unrestricted model (H1)     -6849.914

  Akaike (AIC)                              14088.141
  Bayesian (BIC)                            14209.277
  Sample-size adjusted Bayesian (BIC)       14114.078

Root Mean Square Error of Approximation:
```

```
  RMSEA                                          0.118
  90 Percent confidence interval - lower         0.106
  90 Percent confidence interval - upper         0.130
  P-value RMSEA <= 0.05                          0.000

Standardized Root Mean Square Residual:

  SRMR                                           0.055

Parameter Estimates:

  Standard errors                             Standard
  Information                                 Expected
  Information saturated (h1) model          Structured

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)
  map =~
    ags1              1.000
    ags5              0.774    0.057   13.564    0.000
    ags7              1.100    0.064   17.263    0.000
  mav =~
    ags2              1.000
    ags6              0.974    0.078   12.523    0.000
    ags12             1.039    0.096   10.805    0.000
  pap =~
    ags3              1.000
    ags9              0.853    0.038   22.349    0.000
    ags11             1.103    0.052   21.178    0.000
  pav =~
    ags4              1.000
    ags8              1.599    0.084   19.091    0.000
    ags10             1.525    0.073   20.861    0.000

Covariances:
                   Estimate  Std.Err  z-value  P(>|z|)
  map ~~
    mav               0.709    0.079    9.000    0.000
    pap               0.066    0.060    1.093    0.274
    pav               0.056    0.043    1.289    0.197
  mav ~~
    pap               0.163    0.072    2.265    0.023
```

```
    pav               0.178    0.053    3.355    0.001
  pap ~~
    pav               1.143    0.102   11.236    0.000

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .ags1              0.562    0.047   11.951    0.000
   .ags5              0.486    0.038   12.780    0.000
   .ags7              0.211    0.032    6.602    0.000
   .ags2              1.312    0.102   12.825    0.000
   .ags6              0.469    0.049    9.537    0.000
   .ags12             1.300    0.103   12.669    0.000
   .ags3              0.690    0.059   11.671    0.000
   .ags9              0.386    0.036   10.769    0.000
   .ags11             0.831    0.071   11.639    0.000
   .ags4              0.588    0.045   12.959    0.000
   .ags8              0.815    0.070   11.602    0.000
   .ags10             0.362    0.043    8.423    0.000
    map               0.706    0.083    8.514    0.000
    mav               0.852    0.128    6.655    0.000
    pap               1.648    0.158   10.416    0.000
    pav               0.864    0.094    9.198    0.000
```

```
semPlot::semPaths(h1.fit)
```



That's a lot of outputs. Let's talk about what's here.

2.1.3 Goodness of Fit Statistics

2.1.3.1 Chi-Squared Statistic

The main thing to look at is the chi-squared statistic from the 'User Model', IE the model I, the user, have just fit. I like to think of this as a measure of how different the model's reconstructed correlation matrix looks compared to the actual empirical correlation matrix of the data. So we use this statistic to test the null hypothesis "there is no significant difference between model's reconstructed correlation matrix and the empirical one". So, confusingly, we're actually hoping to *accept* the null hypothesis here. This model returns a value of 328.312 with a vanishingly small p-value, so we reject the null hypothesis, which is bad: it suggests our model isn't doing a good job replicating the empirical correlation matrix.

Here's a quote from Gorsuch (1983) that explains this stuff from the slightly different angle:

> "The test of significance [for a CFA model fit by maximum likelihood] gives a chi-square statistic with the null hypothesis being that all the population covariance has been extracted by the hypothesized number of factors. If the chi-square is significant at the designated probability level, then the residual matrix still has significant covariance in it."

So this chi-squared statistic provides a first look at goodness-of-fit, but @Finch2015 say it is actually not very trustworthy in practice because the null hypothesis is sort of crazy: we want a more permissive test than just whether the model is *perfectly* recreating the empirical correlation matrix.

> "this statistic is not particularly useful in practice because it tests the null hypothesis that [the model-reconstructed correlation matrix is equal to the empirical correlation matrix], which is very restrictive. The test will almost certainly be rejected when the sample size is sufficiently large… In addition, the chi-square test relies on the assumption of multivariate normality of the indicators, which may not be tenable in many situations."

So we're gonna wanna look at statistics other than just chi-squared for goodness-of-fit.

2.1.3.2 Root Mean Squared Error Approximation (RMSEA)

Another one people like to go with is the Root Mean Squared Error Approximation (RMSEA). This statistic takes some math and background to understand, which I'm not going to go over here. I found this document to be the clearest (but also pretty mathy) explanation.

Essentially RMSEA is a weighted sum of the discrepancies between the model's reconstructed correlation matrix and the empirical correlation matrix. But it also does a nice thing where it discounts model complexity and sample size to help us not overfit. Here's the definition:

$$\text{RMSEA} = \sqrt{\frac{\chi^2 - \text{df}}{\text{df}(n-1)}}$$

See how it takes the chi-squared statistic and divides it by degrees of freedom (as a proxy for model complexity) and sample size? This makes for a more conservative measure of goodness-of-fit. Apparently the square-root is used *"to return the index to the same metric as the original standardized parameters"*. I don't really understand that part.

As with the raw chi-squared statistic, we want RMSEA to be small because it is intended as a measure of the distance between the empirical correlation matrix and the model-estimated correlation matrix. According to Finch (2015), people like to say:

- RMSEA <= 0.05 is a 'good fit';
- 0.05 < RMSEA <= 0.08 is an 'ok fit'
- RMSEA > .08 is a 'bad fit'.

Comparative Fit Index (CFI)

2.2 Example 2:

```
dat_ff <- foreign::read.spss('data/finch-and-french/performance.data.sav')

# Seems like I need to only use the first 12 columns I think?
dat_ff <- dat_ff %>%

  as_tibble() %>%

  select(1:12)
```

Next we'll do another example from Finch (2015), from the SEM chapter on page 62. They say it is important to CFA first before testing the relationships between the latent variables a la SEM, because we first want to make sure those latent variables actually seem good. This example uses a different dataset than the previous one.

> [W]e must ascertain whether the proposed model structure is supported by our data, which we will do by fitting a CFA to each of the latent variables…

The authors fit a CFA for each of the latent variables in isolation, namely Mastery, Self-Oriented Perfectionism, and 'ATTC', which seems to mean 'Attention Control'

2.3 Example 3:

Now we'll look at an example from Kline (2011), chapter 13.

Load the data:

```
dat_kline <- read_csv('data/kline/kabc-amos.csv')
```

```
Warning: One or more parsing issues, call `problems()` on your data frame for details,
e.g.:
  dat <- vroom(...)
  problems(dat)
```

```
Rows: 11 Columns: 10
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (2): rowtype_, varname_
dbl (8): HM, NR, WO, GC, Tr, SM, MA, PS
```

```
i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

2.4 Example 4:

Lastly, let's walk through an example from the lavaan documentation

References

Finch, French, W. Holmes. 2015. *Latent Variable Modeling with r.*
Gorsuch, Richard L. 1983. *Factor Analysis, 2nd Edition.*
Kline, Rex B. 2011. *Principles and Practice of Structural Equation Modeling.*