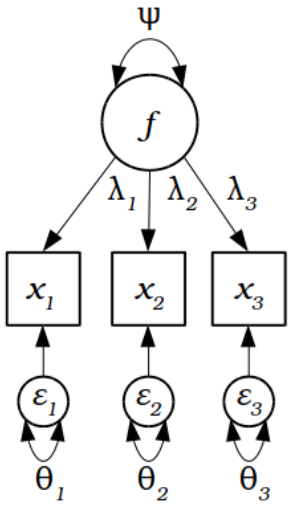
CFA Loading Estimation and Comparison Example

Joel S Steele, PhD

The Common Factor Model

Figure 1: Common factor diagram



Model expectations

Using the tracing rules and our model above in Figure 1, we can analytically specify what the expected elements of the variance covariance matrix should be made up of,

$$\widehat{\Sigma} = \begin{bmatrix} \lambda_1\psi\lambda_1 + \theta_1 & & \\ \lambda_1\psi\lambda_2 & \lambda_2\psi\lambda_2 + \theta_2 & \\ \lambda_1\psi\lambda_3 & \lambda_2\psi\lambda_3 & \lambda_3\psi\lambda_3 + \theta_3 \end{bmatrix}.$$

As specified, the model depicted in Figure 1 is not identified (more on this later). If we include a unit variance identifying constraint, setting $\psi = 1$, the expected covariance matrix becomes,

$$\widehat{\Sigma} = \begin{bmatrix} \lambda_1^2 + \theta_1 & & \\ \lambda_1\lambda_2 & \lambda_2^2 + \theta_2 & \\ \lambda_1\lambda_3 & \lambda_2\lambda_3 & \lambda_3^2 + \theta_3 \end{bmatrix}.$$

Below, we will use these expectations to estimate the model implied loadings from the example covariance matrix in Table 1.

Table 1: Example Covariance Matrix

	V1	V2	V3
V1	4.24	2.70	2.16
V2	2.70	3.45	1.80
V3	2.16	1.80	2.84

With this short 3 item scale we can compute the loadings for items 1, 2, and 3 directly by-hand! This equation comes directly from our expected matrix $\widehat{\Sigma}$ above.

$$\lambda_1^2 = \frac{\sigma_{21} \times \sigma_{31}}{\sigma_{23}} = \frac{\lambda_1\lambda_2 \times \lambda_1\lambda_3}{\lambda_2\lambda_3}$$

Using our equation above we get:

- $\lambda_1 = \sqrt{\frac{2.7 \times 2.16}{1.8}} = 1.8$
- $\lambda_2 = \sqrt{\frac{2.7 \times 1.8}{2.16}} = 1.5$
- $\lambda_3 = \sqrt{\frac{2.16 \times 1.8}{2.7}} = 1.2$

Using our computed loadings, $\{1.8, 1.5, 1.2\}$, we can proceed to estimate the residual or specific variance for each indicator. It's crucial to remember that the total variance for an indicator is a combination of how much the factor(s) explain, and what's left over. In this case we can compute the specific variance from the observed variance values on the diagonal of the covariance matrix in Table 1. These diagonal values are $\{4.24, 3.45, 2.84\}$.

- $\epsilon_1 = \sigma_1^2 - \lambda_1^2 = 4.24 - 1.8^2 = 4.24 - 3.24 = 1.0$
- $\epsilon_2 = \sigma_2^2 - \lambda_2^2 = 3.45 - 1.5^2 = 3.45 - 2.25 = 1.2$
- $\epsilon_3 = \sigma_3^2 - \lambda_3^2 = 2.84 - 1.2^2 = 2.84 - 1.44 = 1.4$

So, at this point, we have estimated what the expected loadings and specific variances should be, based on our model in Figure 1, with a bit of algebra. These estimates are,

λ_1:	1.8	ϵ_1:	1.0
λ_2:	1.5	ϵ_2:	1.2
λ_3:	1.2	ϵ_3:	1.4

Let's see how our computations compare when we run the model using *lavaan* in *R*.

```
# unit variance constraint
testmod = '
  f =~ NA*V1 + V2 + V3
  f ~~ 1*f'
fit1 = cfa(testmod,sample.cov=input, sample.nobs=10000)
tab = parameterEstimates(fit1)[,4:6]
rownames(tab) = rnames
kable(tab,digits=3)
```

	est	se	z
λ_1	1.8	0.018	97.384
λ_2	1.5	0.017	88.418
λ_3	1.2	0.016	76.293
ψ	1.0	0.000	NA
ϵ_1	1.0	0.035	28.489
ϵ_2	1.2	0.028	42.810
ϵ_3	1.4	0.024	57.352

```
# what fit measures do we want
indices = c('chisq','df','pvalue','cfi','rmsea')
fitMeasures(fit1,indices)
```

```
 chisq     df pvalue    cfi  rmsea
     0      0     NA      1      0
```

A few things to note, we included the `NA*V1` portion in the model syntax because otherwise *lavaan* will fix the loading to the first item at one for a factor model by default, and we wanted it free. The `NA` makes it a free estimate. We also included the `f ~~ 1*f` portion to indicate that we want to fix the variance of the factor (the ψ) at 1, which reflects our unit variance identifying constraint.

Notice also, that this model is *JUST* identified. This is reflected in the fact that there are 0 degrees of freedom and the *cfi* and *rmsea* both indicate a perfect fit. Recall that degrees of freedom represent the difference in the number of parameter estimates made, the number of Greek symbols in Figure 1, and the number of unique elements (k) in the covariance matrix in Table 1. The number of unique elements is equal to $\frac{k(k+1)}{2} = \frac{3(4)}{2} = 6$ and there are 6 estimates being made: 3 loadings + 3 specific variances.[1]

As can be seen in the estimates table the computed values for the loadings and the specific variances are exactly what we computed by hand earlier!

Equivalent fit, different scaling

Now, the question often comes up regarding what will happen if the loading to the first item is fixed at one compared to fixing the variance of the factor at one? Below, we fit such a model, keep in mind that *lavaan* imposes the unit loading constraint by default.

```
# unit loading constraint 1st var
testmod2 = '
  f =~ V1 + V2 + V3
  f ~~ f'
fit2 = cfa(testmod2,sample.cov=input, sample.nobs=10000)
tab2 = parameterestimates(fit2)[,4:6]
rownames(tab2) = rnames
kable(tab2,digits=3)
```

	est	se	z
λ_1	1.000	0.000	NA
λ_2	0.833	0.011	77.602
λ_3	0.667	0.009	71.356
ψ	3.240	0.067	48.692
ϵ_1	1.000	0.035	28.489
ϵ_2	1.200	0.028	42.810
ϵ_3	1.400	0.024	57.352

```
fitMeasures(fit2,indices)
```

```
 chisq     df pvalue    cfi  rmsea
     0      0     NA      1      0
```

The first thing to note is that the degrees of freedom and model fit are exactly the same. Why? Well because we've just swapped one constraint for another.

Next thing worth noting is that there are some differences in the model parameter estimates. For example there is now an estimate for the variance of the factor $\psi = 3.240$ which comes from the fact that we have set the scale of the factor to be equal to the first item through use of the unit loading constraint. Remember that in our earlier model where the factor variance was constrained to equal 1, our loadings were estimated as $\lambda = \{1.8, 1.5, 1.2\}$, well the square of the first loading is, $1.8^2 = 3.24$, which is now the estimate of the factor variance.

[1] Remember that we constrained the factor variance $\psi = 1$.

Also, notice that the estimates for the other loadings also differ from our earlier model. This is again reflective of the scaling of the factor that the unit loading constraint provides. Notice that if we divide our earlier estimates by our constrained loading we get our new *rescaled* estimates,

$$\frac{\lambda_i}{\lambda_1} = \frac{1.8, 1.5, 1.2}{1.8} = \{1.00, 0.833, 0.667\},$$

which match the estimates listed above.

Just to illustrate this scaling again, we will fit the model again but this time we will constrain the last loading to one.

```
# unit loading constraint last var
testmod2b = '
  f =~ NA*V1 + V2 + 1*V3
  f ~~ f'
fit2b = cfa(testmod2b,sample.cov=input, sample.nobs=10000)
tab2b = parameterestimates(fit2b)[,4:6]
rownames(tab2b) = rnames
kable(tab2b,digits=3)
```

	est	se	z
λ_1	1.50	0.021	71.356
λ_2	1.25	0.018	71.243
λ_3	1.00	0.000	NA
ψ	1.44	0.038	38.147
ϵ_1	1.00	0.035	28.489
ϵ_2	1.20	0.028	42.810
ϵ_3	1.40	0.024	57.352

```
fitMeasures(fit2b,indices)
```

```
 chisq     df pvalue    cfi  rmsea
     0      0     NA      1      0
```

Notice again, the χ^2 is the same as are the degrees of freedom and other indications of fit. The variance of the factor is now equal to the square of the last loading $1.2^2 = 1.44$, and the estimated loadings are also re-scaled by the last indicator,

$$\frac{\lambda_i}{\lambda_3} = \frac{1.8, 1.5, 1.2}{1.2} = \{1.50, 1.25, 1.00\}.$$

Hopefully, this helps clear up any ambiguity there may be about setting the scale of the factor with either the unit loading or unit variance constraints.

Adding non-identifying constraints.

Next we move on to showing how degrees of freedom are impacted by additional constraints. Up to this point we've just been concerned with identifying constraints, namely ones that set the scale of the factor in some way, and have been looking at *JUST* identified models, wherein the number of estimated parameters is equal to the number of unique elements in the covariance matrix.

Below, we will see that by setting parameters in the model equal to each other, through use of labels in *lavaan*, we will gain some degrees of freedom.

```
# unit variance constraint
testmod3 = '
  f =~ NA*V1 + V2 + V3
  f ~~ 1*f
  V1 ~~ u*V1
  V2 ~~ u*V2
  V3 ~~ u*V3'
fit3 = cfa(testmod3,sample.cov=input, sample.nobs=10000)
tab3 = parameterestimates(fit3)[,5:7]
rownames(tab3) = rnames
kable(tab3,digits=3)
```

	est	se	z
λ_1	1.752	0.017	101.946
λ_2	1.502	0.016	94.016
λ_3	1.238	0.015	83.515
ψ	1.000	0.000	NA
ϵ_1	1.224	0.012	100.000
ϵ_2	1.224	0.012	100.000
ϵ_3	1.224	0.012	100.000

```
fitMeasures(fit3,indices)
```

```
 chisq     df pvalue    cfi  rmsea
91.593  2.000  0.000  0.993  0.067
```

In the model above we retain the unit variance constraint to set the scale of our factor. This is a necessary step, factors always need a set scale regardless of the number of degrees of freedom. Additionally, we have set the specific variances among the indicators, or manifest variable, to be equal via the same label `u` being used as shown below

```
  V1 ~~ u*V1
  V2 ~~ u*V2
  V3 ~~ u*V3
```

What this amounts to is that the model no longer has to estimate three specific variances, instead it estimates one and sets the other to be equal to it. As can be see in the fit statistics we've gained $3 - 1 = 2$ degrees of freedom as a result.

Comparisons

	m1.est	m1.se	m2.est	m2.se	m2b.est	m2b.se	m3.est	m3.se
λ_1	1.8	0.018	1.00	0.000	1.50	0.021	1.75	0.017
λ_2	1.5	0.017	0.83	0.011	1.25	0.018	1.50	0.016
λ_3	1.2	0.016	0.67	0.009	1.00	0.000	1.24	0.015
ψ	1.0	0.000	3.24	0.067	1.44	0.038	1.00	0.000
ϵ_1	1.0	0.035	1.00	0.035	1.00	0.035	1.22	0.012
ϵ_2	1.2	0.028	1.20	0.028	1.20	0.028	1.22	0.012
ϵ_3	1.4	0.024	1.40	0.024	1.40	0.024	1.22	0.012