Examples 3.3 and 3.4 - Confirmatory factor analysis of five substests of the Wechsler Intelligence Scale for Children - IV (WISC-IV)

Part I: Single Factor Model

First, we load the lavaan package and read in the data:

```
library("lavaan")
```

```
wisc4.cor <- lav_matrix_lower2full(c(1.0,
                                    .72, 1.0,
                                    .64, .63, 1.0,
                                    .51, .48, .37, 1.0,
                                    .37, .38, .38, .38, 1))
colnames(wisc4.cor) <- rownames(wisc4.cor) <- c("Information", "Similarities",
                                                "Word.Reasoning", "Matrix.Reasoning",
                                                "Picture.Concepts")
wisc4.sd <- c(3.01, 3.03, 2.99, 2.89, 2.98)
names(wisc4.sd) <- colnames(wisc4.cor)
wisc4.cov <- cor2cov(wisc4.cor, wisc4.sd)
```

Marker variable identification (default)

Using the first indicator for identification of the scale of the latent variable is the default in lavaan. In that case, we would specify and fit the model as follows:

```
wisc4.model <- '
  g =~ Information + Similarities + Word.Reasoning + Matrix.Reasoning +
        Picture.Concepts
'
wisc4.fit <- cfa(wisc4.model, sample.cov = wisc4.cov, sample.nobs = 550)
summary(wisc4.fit, standardized = TRUE)
```

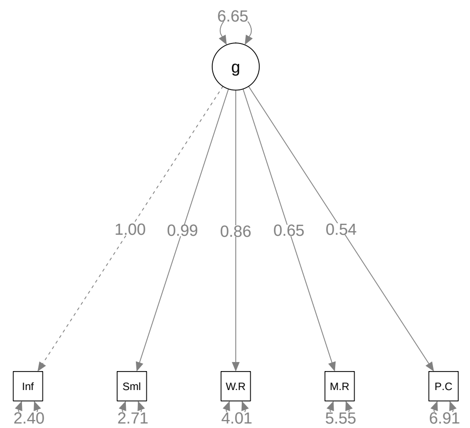
```
## lavaan 0.6-5 ended normally after 30 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of free parameters                         10
##
##   Number of observations                           550
##
## Model Test User Model:
##
##   Test statistic                                26.775
##   Degrees of freedom                                 5
##   P-value (Chi-square)                           0.000
##
## Parameter Estimates:
##
```

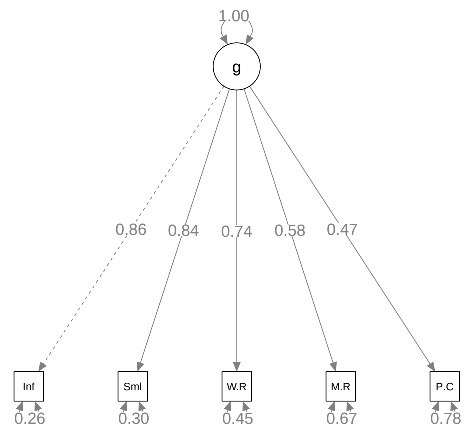
```
##   Information                                 Expected
##   Information saturated (h1) model          Structured
##   Standard errors                             Standard
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   g =~
##     Information       1.000                               2.578    0.857
##     Similarities      0.985    0.045   21.708    0.000    2.541    0.839
##     Word.Reasoning    0.860    0.045   18.952    0.000    2.217    0.742
##     Matrix.Reasnng    0.647    0.047   13.896    0.000    1.669    0.578
##     Picture.Cncpts    0.542    0.050   10.937    0.000    1.398    0.470
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .Information       2.395    0.250    9.587    0.000    2.395    0.265
##    .Similarities      2.709    0.258   10.482    0.000    2.709    0.296
##    .Word.Reasoning    4.009    0.295   13.600    0.000    4.009    0.449
##    .Matrix.Reasnng    5.551    0.360   15.400    0.000    5.551    0.666
##    .Picture.Cncpts    6.909    0.434   15.922    0.000    6.909    0.779
##     g                 6.648    0.564   11.788    0.000    1.000    1.000
```

```r
library("semPlot")
```

```
## Warning: package 'semPlot' was built under R version 4.0.2
```

```r
par(mfrow = c(1, 2))
semPaths(wisc4.fit, whatLabels = "est", edge.label.cex = 1.2)
semPaths(wisc4.fit, whatLabels = "std", edge.label.cex = 1.2)
```





Standardized latent variable identification

We can also identify the scale of the latent variable(s) by fixing their variance to one. This can be done in **lavaan** in two ways:

```r
wisc4.fit.std1 <- cfa(wisc4.model, sample.cov = wisc4.cov, sample.nobs = 550,
                      std.lv = TRUE)
ests <- c("lhs", "op", "rhs", "est", "pvalue", "std.lv", "std.all")
parameterEstimates(wisc4.fit.std1, standardized = TRUE)[ests]
```

```
##                lhs op               rhs   est pvalue std.lv std.all
## 1                g =~       Information 2.578      0  2.578   0.857
## 2                g =~      Similarities 2.541      0  2.541   0.839
## 3                g =~    Word.Reasoning 2.217      0  2.217   0.742
## 4                g =~  Matrix.Reasoning 1.669      0  1.669   0.578
## 5                g =~  Picture.Concepts 1.398      0  1.398   0.470
## 6      Information ~~       Information 2.395      0  2.395   0.265
## 7     Similarities ~~      Similarities 2.709      0  2.709   0.296
## 8   Word.Reasoning ~~    Word.Reasoning 4.009      0  4.009   0.449
## 9 Matrix.Reasoning ~~  Matrix.Reasoning 5.551      0  5.551   0.666
## 10 Picture.Concepts ~~ Picture.Concepts 6.909      0  6.909   0.779
## 11               g ~~                 g 1.000     NA  1.000   1.000
```

```
wisc4.model.std2 <- '
    g =~ NA*Information + Similarities + Word.Reasoning + Matrix.Reasoning +
         Picture.Concepts
    g ~~ 1*g
'
wisc4.fit.std2 <- cfa(wisc4.model.std2, sample.cov = wisc4.cov,
                      sample.nobs = 550)
parameterEstimates(wisc4.fit.std2, standardized = TRUE)[ests]
```

```
##                lhs op               rhs   est pvalue std.lv std.all
## 1                g =~       Information 2.578      0  2.578   0.857
## 2                g =~      Similarities 2.541      0  2.541   0.839
## 3                g =~    Word.Reasoning 2.217      0  2.217   0.742
## 4                g =~  Matrix.Reasoning 1.669      0  1.669   0.578
## 5                g =~  Picture.Concepts 1.398      0  1.398   0.470
## 6                g ~~                 g 1.000     NA  1.000   1.000
## 7      Information ~~       Information 2.395      0  2.395   0.265
## 8     Similarities ~~      Similarities 2.709      0  2.709   0.296
## 9   Word.Reasoning ~~    Word.Reasoning 4.009      0  4.009   0.449
## 10 Matrix.Reasoning ~~ Matrix.Reasoning 5.551      0  5.551   0.666
## 11 Picture.Concepts ~~ Picture.Concepts 6.909      0  6.909   0.779
```

Note I used the `parameterEstimates()` function here, and not the `summary()` function, in order to save space. It would have given me the same results.

Note that the `std.all` column provides completely standardized estimates, which can be interpreted as correlation coefficients (i.e., take values between -1 and 1). Note also that the `std.lv` columns provide a solution where the latent factor is assumed to have variance and standard deviation equal to 1. In general, for interpretation, the `std.all` column provides the most useful results.

Which subtests have low loadings and/or high uniquenesses (error variances)?

Part II: Parameter matrices

We can extract the estimated parameters as matrices:

```r
inspect(wisc4.fit, what = "est")
```

```
## $lambda
##                     g
## Information     1.000
## Similarities    0.985
## Word.Reasoning  0.860
## Matrix.Reasoning 0.647
## Picture.Concepts 0.542
##
## $theta
##                  Infrmt Smlrts Wrd.Rs Mtrx.R Pctr.C
## Information      2.395
## Similarities     0.000  2.709
## Word.Reasoning   0.000  0.000  4.009
## Matrix.Reasoning 0.000  0.000  0.000  5.551
## Picture.Concepts 0.000  0.000  0.000  0.000  6.909
##
## $psi
##   g
## g 6.648
```

Part III: Evaluating Model Fit

Let's inspect the model fit for the unidimensional model:

```
fitmeasures(wisc4.fit)
```

```
##                npar                fmin               chisq                  df 
##              10.000               0.024              26.775               5.000 
##              pvalue      baseline.chisq         baseline.df     baseline.pvalue 
##               0.000            1073.427              10.000               0.000 
##                 cfi                 tli                nnfi                 rfi 
##               0.980               0.959               0.959               0.950 
##                 nfi                pnfi                 ifi                 rni 
##               0.975               0.488               0.980               0.980 
##                logl   unrestricted.logl                 aic                 bic 
##           -6378.678           -6365.291           12777.357           12820.456 
##              ntotal                bic2               rmsea      rmsea.ci.lower 
##             550.000           12788.712               0.089               0.058 
##      rmsea.ci.upper        rmsea.pvalue                 rmr          rmr_nomean 
##               0.123               0.022               0.298               0.298 
##                srmr        srmr_bentler srmr_bentler_nomean                crmr 
##               0.034               0.034               0.034               0.042 
##         crmr_nomean          srmr_mplus   srmr_mplus_nomean               cn_05 
##               0.042               0.034               0.034             228.408 
##               cn_01                 gfi                agfi                pgfi 
##             310.899               0.982               0.947               0.327 
##                 mfi                ecvi 
##               0.980               0.085
```

```
measures <- c("chisq", "df", "pvalue", "cfi", "rmsea", "srmr")
fitmeasures(wisc4.fit, measures)
```

```
##  chisq     df pvalue    cfi  rmsea   srmr 
## 26.775  5.000  0.000  0.980  0.089  0.034
```

Does the model fit well according to the chi-square? Is that to be expected with this sample size? Does the model fit well according to CFI? SRMR? RMSEA?

Part IV: Inspecting Model Misfit

Let's inspect modification indices:

```
modindices(wisc4.fit, sort=TRUE)
```

```
##                   lhs op               rhs     mi    epc sepc.lv sepc.all sepc.nox
## 21 Matrix.Reasoning ~~ Picture.Concepts 14.157  1.058   1.058    0.171    0.171
## 19   Word.Reasoning ~~ Matrix.Reasoning  8.931 -0.710  -0.710   -0.151   -0.151
## 15      Information ~~ Picture.Concepts  5.493 -0.565  -0.565   -0.139   -0.139
## 20   Word.Reasoning ~~ Picture.Concepts  2.029  0.365   0.365    0.069    0.069
## 14      Information ~~ Matrix.Reasoning  1.447  0.280   0.280    0.077    0.077
## 18     Similarities ~~ Picture.Concepts  0.838 -0.223  -0.223   -0.051   -0.051
## 16     Similarities ~~   Word.Reasoning  0.791  0.242   0.242    0.073    0.073
## 13      Information ~~   Word.Reasoning  0.279  0.147   0.147    0.047    0.047
## 17     Similarities ~~ Matrix.Reasoning  0.147 -0.089  -0.089   -0.023   -0.023
## 12      Information ~~     Similarities  0.010  0.034   0.034    0.013    0.013
```

What changes to the model are suggested by the modification indices?

Let's also look at the difference between the model-implied and sample covariances:

```
fitted(wisc4.fit)$cov # model-implied covariances
```

```
##                  Infrmt Smlrts Wrd.Rs Mtrx.R Pctr.C
## Information      9.044
## Similarities     6.551  9.164
## Word.Reasoning   5.716  5.633  8.924
## Matrix.Reasoning 4.303  4.241  3.700  8.337
## Picture.Concepts 3.606  3.553  3.100  2.334  8.864
```

```
residuals(wisc4.fit)$cov # unstandardized residuals
```

```
##                  Infrmt Smlrts Wrd.Rs Mtrx.R Pctr.C
## Information       0.000
## Similarities      0.003  0.000
## Word.Reasoning    0.033  0.064  0.000
## Matrix.Reasoning  0.125 -0.045 -0.509  0.000
## Picture.Concepts -0.293 -0.128  0.280  0.933  0.000
```

```
residuals(wisc4.fit, type="cor")$cor # standardized residuals
```

```
## NULL
```

As a rule of thumb, standardized residuals with absolute values $> .1$ are substantial. Can you think of ways to reduce residual correlations with absolute values $> .1$?

Additional: Computing the residual covariance matrix from model matrices

The matrices of parameter estimates of our CFA model are as follows:

```
mats <- inspect(wisc4.fit, "est")
mats
```

```
## $lambda
##                      g
## Information      1.000
## Similarities     0.985
## Word.Reasoning   0.860
## Matrix.Reasoning 0.647
```

```
## Picture.Concepts 0.542
##
## $theta
##                   Infrmt Smlrts Wrd.Rs Mtrx.R Pctr.C
## Information       2.395
## Similarities      0.000  2.709
## Word.Reasoning    0.000  0.000  4.009
## Matrix.Reasoning  0.000  0.000  0.000  5.551
## Picture.Concepts  0.000  0.000  0.000  0.000  6.909
##
## $psi
##   g
## g 6.648
```

Because we only have a measurement model (i.e., no regressions), the model-implied covariance matrix is given by:

$$\hat{\Sigma} = \Lambda \Psi \Lambda^T + \Theta$$

We can compute $\hat{\Sigma}$ ourselves (note that in `R`, `%*%` is used for matrix multiplication, `t()` gives the transpose and `solve()` gives the inverse of a matrix):

```
sigma_hat <- mats$lambda %*% mats$psi %*% t(mats$lambda) + mats$theta
sigma_hat
```

```
##                   Infrmt Smlrts Wrd.Rs Mtrx.R Pctr.C
## Information       9.044
## Similarities      6.551  9.164
## Word.Reasoning    5.716  5.633  8.924
## Matrix.Reasoning  4.303  4.241  3.700  8.337
## Picture.Concepts  3.606  3.553  3.100  2.334  8.864
```

We could have extracted the model-implied covariance matrix directly using function `fitted()`:

```
fitted(wisc4.fit)
```

```
## $cov
##                   Infrmt Smlrts Wrd.Rs Mtrx.R Pctr.C
## Information       9.044
## Similarities      6.551  9.164
## Word.Reasoning    5.716  5.633  8.924
## Matrix.Reasoning  4.303  4.241  3.700  8.337
## Picture.Concepts  3.606  3.553  3.100  2.334  8.864
```

With ML estimation, (co)variances are computed using N instead of $(N-1)$ as the divisor. This is because the formula for the sample (co)variance has $(N-1)$ as the divisor, while the formula for the population (co)variance has N as the divisor. Our original (co)variance matrix was a sample (co)variance matrix, which was computed using $(N-1)$ as the divisor. The sample size was $N = 550$. To compute the residuals, we first have to rescale it using $\frac{N-1}{N}$:

```
wisc4.cov*(549/550)
```

```
##                   Information Similarities Word.Reasoning Matrix.Reasoning
## Information          9.043627     6.554677       5.749463         4.428373
## Similarities         6.554677     9.164207       5.697234         4.195574
## Word.Reasoning       5.749463     5.697234       8.923845         3.191394
## Matrix.Reasoning     4.428373     4.195574       3.191394         8.336914
## Picture.Concepts     3.312792     3.424934       3.379720         3.266686
```

```
##                   Picture.Concepts
## Information               3.312792
## Similarities              3.424934
## Word.Reasoning            3.379720
## Matrix.Reasoning          3.266686
## Picture.Concepts          8.864254
```

From the (co)variance matrix, we substract the model-implied covariance matrix $\hat{\Sigma}$, which gives us the residual matrix:

```r
wisc4.cov*(549/550) - sigma_hat
```

```
##                   Infrmt Smlrts Wrd.Rs Mtrx.R Pctr.C
## Information        0.000
## Similarities       0.003  0.000
## Word.Reasoning     0.033  0.064  0.000
## Matrix.Reasoning   0.125 -0.045 -0.509  0.000
## Picture.Concepts  -0.293 -0.128  0.280  0.933  0.000
```

Note that all variances are perfectly replicated, which is possible because all error variances were freely estimated in the model. Non-zero residuals are only observed for the covariances between the subscales.

Note also that our manually computed result is identical to the result of the `residuals()` function:

```r
residuals(wisc4.fit, type = "raw")
```

```
## $type
## [1] "raw"
##
## $cov
##                   Infrmt Smlrts Wrd.Rs Mtrx.R Pctr.C
## Information        0.000
## Similarities       0.003  0.000
## Word.Reasoning     0.033  0.064  0.000
## Matrix.Reasoning   0.125 -0.045 -0.509  0.000
## Picture.Concepts  -0.293 -0.128  0.280  0.933  0.000
```

Part V: Adjusting the Model

In the book, Beaujean decided to improve the model by including Verbal and Fluid intelligence factors:

```
wisc4.model2 <- '
  V =~ Information + Similarities + Word.Reasoning
  F =~ Matrix.Reasoning + Picture.Concepts
'
wisc4.fit.2 <- cfa(wisc4.model2, sample.cov = wisc4.cov,
                   sample.nobs = 550)
summary(wisc4.fit.2, standardized = TRUE)
```

```
## lavaan 0.6-5 ended normally after 44 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of free parameters                         11
##
##   Number of observations                           550
##
## Model Test User Model:
##
##   Test statistic                                12.687
##   Degrees of freedom                                 4
##   P-value (Chi-square)                           0.013
##
## Parameter Estimates:
##
##   Information                                 Expected
##   Information saturated (h1) model          Structured
##   Standard errors                             Standard
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   V =~
##     Information       1.000                               2.587    0.860
##     Similarities      0.984    0.046   21.625    0.000    2.545    0.841
##     Word.Reasoning    0.858    0.045   18.958    0.000    2.219    0.743
##   F =~
##     Matrix.Reasnng    1.000                               1.989    0.689
##     Picture.Cncpts    0.825    0.085    9.747    0.000    1.642    0.552
##
## Covariances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   V ~~
##     F                 4.233    0.399   10.604    0.000    0.823    0.823
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .Information       2.352    0.253    9.295    0.000    2.352    0.260
##    .Similarities      2.685    0.261   10.282    0.000    2.685    0.293
##    .Word.Reasoning    4.000    0.295   13.555    0.000    4.000    0.448
##    .Matrix.Reasnng    4.380    0.458    9.557    0.000    4.380    0.525
##    .Picture.Cncpts    6.168    0.451   13.673    0.000    6.168    0.696
##     V                 6.692    0.567   11.807    0.000    1.000    1.000
```

```
##     F                 3.957    0.569    6.960    0.000    1.000    1.000
```

As a rule-of-thumb, a LV model needs at least 3 indicators to be identified. How come the above LV model is identified?

Also, not that the covariance between the two latent factors included in the model was estimated by default, even though we did not specify it in the model syntax.

We could also include a structural model, where we assume a causal relationship between the two types of intelligence:

```
wisc4.model3 <- '
  V =~ Information + Similarities + Word.Reasoning
  F =~ Matrix.Reasoning + Picture.Concepts
  V ~ F
'
wisc4.fit.3 <- cfa(wisc4.model3, sample.cov = wisc4.cov, sample.nobs = 550)
```

Or, we could have done exactly what the modification indices suggested above:

```
wisc4.model4 <- '
  g =~ Information + Similarities + Word.Reasoning + Matrix.Reasoning +
        Picture.Concepts
  Matrix.Reasoning ~~ Picture.Concepts
'
wisc4.fit.4 <- cfa(wisc4.model4, sample.cov = wisc4.cov,
                   sample.nobs = 550)
```

Which of the models fits best?

```
fitMeasures(wisc4.fit, measures)
```

```
##  chisq     df pvalue    cfi  rmsea   srmr
## 26.775  5.000  0.000  0.980  0.089  0.034
```

```
fitMeasures(wisc4.fit.2, measures)
```

```
##  chisq     df pvalue    cfi  rmsea   srmr
## 12.687  4.000  0.013  0.992  0.063  0.019
```

```
fitMeasures(wisc4.fit.3, measures)
```

```
##  chisq     df pvalue    cfi  rmsea   srmr
## 12.687  4.000  0.013  0.992  0.063  0.019
```

```
fitMeasures(wisc4.fit.4, measures)
```

```
##  chisq     df pvalue    cfi  rmsea   srmr
## 12.687  4.000  0.013  0.992  0.063  0.019
```

The last three models have better fit than the first. But note that the last three models are equivalent: They have an identical number of estimated parameters and identical model fit. They also have identical residual (co)variance matrices:

```
residuals(wisc4.fit, type = "cor")
```

```
## $type
## [1] "cor.bollen"
##
## $cov
##                  Infrmt Smlrts Wrd.Rs Mtrx.R Pctr.C
```

```
## Information       0.000
## Similarities      0.000  0.000
## Word.Reasoning    0.004  0.007  0.000
## Matrix.Reasoning  0.014 -0.005 -0.059  0.000
## Picture.Concepts -0.033 -0.014  0.031  0.109  0.000
```

```r
residuals(wisc4.fit.2, type = "cor")
```

```
## $type
## [1] "cor.bollen"
##
## $cov
##                  Infrmt Smlrts Wrd.Rs Mtrx.R Pctr.C
## Information       0.000
## Similarities     -0.003  0.000
## Word.Reasoning    0.001  0.005  0.000
## Matrix.Reasoning  0.023  0.003 -0.051  0.000
## Picture.Concepts -0.020 -0.001  0.043  0.000  0.000
```

```r
residuals(wisc4.fit.3, type = "cor")
```

```
## $type
## [1] "cor.bollen"
##
## $cov
##                  Infrmt Smlrts Wrd.Rs Mtrx.R Pctr.C
## Information       0.000
## Similarities     -0.003  0.000
## Word.Reasoning    0.001  0.005  0.000
## Matrix.Reasoning  0.023  0.003 -0.051  0.000
## Picture.Concepts -0.020 -0.001  0.043  0.000  0.000
```

```r
residuals(wisc4.fit.4, type = "cor")
```

```
## $type
## [1] "cor.bollen"
##
## $cov
##                  Infrmt Smlrts Wrd.Rs Mtrx.R Pctr.C
## Information       0.000
## Similarities     -0.003  0.000
## Word.Reasoning    0.001  0.005  0.000
## Matrix.Reasoning  0.023  0.003 -0.051  0.000
## Picture.Concepts -0.020 -0.001  0.043  0.000  0.000
```

In other words, the data cannot discriminate between the last three models. Only the researcher can, by using theory and interpreting the model.