Example 6.2 - Ordered-categorical indicator variables, Parts II-IV

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

Part II: IRT approach (maximum likelihood estimation)

Using R package **ltm**, we can perform a similar analysis, but now using ML estimation.

```
library("ltm")
lsat.IRT <- ltm(lsat6 ~ z1)
summary(lsat.IRT)
```

```
##
## Call:
## ltm(formula = lsat6 ~ z1)
##
## Model Summary:
##    log.Lik      AIC      BIC
##  -2466.653 4953.307 5002.384
##
## Coefficients:
##              value std.err  z.vals
## Dffclt.Q1 -3.3597  0.8669 -3.8754
## Dffclt.Q2 -1.3696  0.3073 -4.4565
## Dffclt.Q3 -0.2799  0.0997 -2.8083
## Dffclt.Q4 -1.8659  0.4341 -4.2982
## Dffclt.Q5 -3.1236  0.8700 -3.5904
## Dscrmn.Q1  0.8254  0.2581  3.1983
## Dscrmn.Q2  0.7229  0.1867  3.8721
## Dscrmn.Q3  0.8905  0.2326  3.8281
## Dscrmn.Q4  0.6886  0.1852  3.7186
## Dscrmn.Q5  0.6575  0.2100  3.1306
##
## Integration:
## method: Gauss-Hermite
## quadrature points: 21
##
## Optimization:
## Convergence: 0
## max(|grad|): 0.024
## quasi-Newton: BFGS
```

The difficulty parameters reveal a similar ordering of item difficulty as the thresholds we estimated earlier. The discrimination parameters reveal a similar (but not completely identical) ordering of indicator strength as the loadings we estimated earlier.

Part III: Comparing the fit of the Rasch (1PL) and 2PL model

In the Rasch model, the probability of a correct answer is a function of the subject's ability and the item's difficulty:

$$p(Y = 1|\theta_j, \beta_i) = \frac{e^{\theta_j - \beta_i}}{1 + e^{\theta_j - \beta_i}}$$

where θ_j is the ability of person j, and β_i is the difficulty of item i.

In the 2PL model, the probability of a correct answer is additionaly determined by the item's discriminatory power:

$$p(Y = 1|\theta_j, \beta_i, \alpha_i) = \frac{e^{\alpha_i(\theta_j - \beta_i)}}{1 + e^{\alpha_i(\theta_j - \beta_i)}}$$

where α_i is the discrimination parameter of item i.

The CFA model we fit previously was actually a 2-parameter model, because it estimated difficulties (or thresholds) for all items, as well as discrimination parameters (or loadings):

```
model.CFA <- '
  Theta =~ Q1 + Q2 + Q3 + Q4 + Q5
'
fit.CFA <- cfa(model.CFA, data = data.frame(lsat6), ordered = paste0("Q", 1:5))
```

We can empirically decide between the Rasch and 2PL model, by fitting both models to the data, and testing the difference in model fit.

We can do that using DWLS estimation in **lavaan**:

```
model.rasch <- '
  Theta =~ lambda*Q1 + lambda*Q2 + lambda*Q3 + lambda*Q4 + lambda*Q5
'
```

Note that I pre-multiplied all items with `lambda`. As a result, every item's loading will receive the same label, and all loadings will have the same estimated value. In effect, this applies an equality restriction on the item loadings.

We fit the model to the data and inspect the results:

```
fit.rasch <- cfa(model.rasch, data = data.frame(lsat6), ordered = paste0("Q", 1:5))
summary(fit.rasch, standardized = TRUE, fit.measures = TRUE)
```

```
## lavaan 0.6.15 ended normally after 2 iterations
##
##   Estimator                                         DWLS
##   Optimization method                             NLMINB
##   Number of model parameters                           6
##
##   Number of observations                            1000
##
## Model Test User Model:
##                                                 Standard      Scaled
```

```
##   Test Statistic                                 4.943       5.350
##   Degrees of freedom                                 9           9
##   P-value (Chi-square)                           0.839       0.803
##   Scaling correction factor                                  0.961
##   Shift parameter                                            0.209
##     simple second-order correction
##
## Model Test Baseline Model:
##
##   Test statistic                                67.171      65.104
##   Degrees of freedom                                10          10
##   P-value                                        0.000       0.000
##   Scaling correction factor                                  1.038
##
## User Model versus Baseline Model:
##
##   Comparative Fit Index (CFI)                    1.000       1.000
##   Tucker-Lewis Index (TLI)                       1.079       1.074
##
##   Robust Comparative Fit Index (CFI)                         1.000
##   Robust Tucker-Lewis Index (TLI)                            1.097
##
## Root Mean Square Error of Approximation:
##
##   RMSEA                                          0.000       0.000
##   90 Percent confidence interval - lower         0.000       0.000
##   90 Percent confidence interval - upper         0.021       0.023
##   P-value H_0: RMSEA <= 0.050                    1.000       1.000
##   P-value H_0: RMSEA >= 0.080                    0.000       0.000
##
##   Robust RMSEA                                               0.000
##   90 Percent confidence interval - lower                     0.000
##   90 Percent confidence interval - upper                     0.060
##   P-value H_0: Robust RMSEA <= 0.050                         0.915
##   P-value H_0: Robust RMSEA >= 0.080                         0.012
##
## Standardized Root Mean Square Residual:
##
##   SRMR                                           0.041       0.041
##
## Parameter Estimates:
##
##   Standard errors                           Robust.sem
##   Information                                 Expected
##   Information saturated (h1) model        Unstructured
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   Theta =~
##     Q1      (lmbd)    1.000                               0.400    0.400
##     Q2      (lmbd)    1.000                               0.400    0.400
##     Q3      (lmbd)    1.000                               0.400    0.400
##     Q4      (lmbd)    1.000                               0.400    0.400
##     Q5      (lmbd)    1.000                               0.400    0.400
```

```
##
## Intercepts:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .Q1                0.000                                0.000    0.000
##    .Q2                0.000                                0.000    0.000
##    .Q3                0.000                                0.000    0.000
##    .Q4                0.000                                0.000    0.000
##    .Q5                0.000                                0.000    0.000
##     Theta             0.000                                0.000    0.000
##
## Thresholds:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##     Q1|t1            -1.433    0.059  -24.431    0.000   -1.433   -1.433
##     Q2|t1            -0.550    0.042  -13.133    0.000   -0.550   -0.550
##     Q3|t1            -0.133    0.040   -3.349    0.001   -0.133   -0.133
##     Q4|t1            -0.716    0.044  -16.430    0.000   -0.716   -0.716
##     Q5|t1            -1.126    0.050  -22.395    0.000   -1.126   -1.126
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .Q1                0.840                                0.840    0.840
##    .Q2                0.840                                0.840    0.840
##    .Q3                0.840                                0.840    0.840
##    .Q4                0.840                                0.840    0.840
##    .Q5                0.840                                0.840    0.840
##     Theta             0.160    0.025    6.341    0.000    1.000    1.000
##
## Scales y*:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##     Q1                1.000                                1.000    1.000
##     Q2                1.000                                1.000    1.000
##     Q3                1.000                                1.000    1.000
##     Q4                1.000                                1.000    1.000
##     Q5                1.000                                1.000    1.000
```

We see good model fit according to all indices. Note that we have more degrees of freedom, because we estimated less parameters than in the previous model (Rasch model estimates 1 loading, the earlier model estimated 5 separate loadings for the items). We see that the standardized loadings are substantial and significant. The latent variable (`theta`) has significant variance. The ordering of item difficulties remained the same.

So should we prefer the more parsimoneous Rasch model, or the more complex 2PL model? Although this is also a matter of personal preference (parsimonity vs. complexity), we can also decide on statistical grounds, by comparing the fit indices:

```r
fitinds <- c("chisq.scaled", "df", "pvalue.scaled", "cfi.scaled",
             "rmsea.scaled", "srmr")
fitMeasures(fit.CFA, fitinds)
```

```
##  chisq.scaled            df pvalue.scaled    cfi.scaled  rmsea.scaled
##         4.740         5.000         0.448         1.000         0.000
##          srmr
##         0.036
```

```r
fitMeasures(fit.rasch, fitinds)
```

```
##  chisq.scaled            df pvalue.scaled    cfi.scaled  rmsea.scaled
##         5.350         9.000         0.803         1.000         0.000
##          srmr
##         0.041
```

Both models show excellent fit. Although χ^2 and SRMR indicate closer fit to the data for the 2PL model, the *df* indicate more parsimonity for the 1PL model.

We can also statistically test the difference in model fit using a likelihood-ratio test:

```r
lavTestLRT(fit.rasch, fit.CFA)
```

```
##
## Scaled Chi-Squared Difference Test (method = "satorra.2000")
##
## lavaan NOTE:
##     The "Chisq" column contains standard test statistics, not the
##     robust test that should be reported per model. A robust difference
##     test is a function of two standard (not robust) statistics.
##
##           Df AIC BIC  Chisq Chisq diff Df diff Pr(>Chisq)
## fit.CFA    5         4.0511
## fit.rasch  9         4.9433     0.8764       4     0.9279
```

The likeihood ratio test indicates no signficiant difference in model fit between the 1- and 2PL model. In that case, we prefer the more parsimonious model: The Rasch (1PL) model.

We could do the same comparison for the ML-estimated models:

```r
lsat.IRT.rasch <- rasch(lsat6)
summary(lsat.IRT.rasch)
```

```
##
## Call:
## rasch(data = lsat6)
##
## Model Summary:
##    log.Lik      AIC      BIC
##  -2466.938 4945.875 4975.322
##
## Coefficients:
##             value std.err   z.vals
## Dffclt.Q1 -3.6153  0.3266 -11.0680
## Dffclt.Q2 -1.3224  0.1422  -9.3009
## Dffclt.Q3 -0.3176  0.0977  -3.2518
## Dffclt.Q4 -1.7301  0.1691 -10.2290
## Dffclt.Q5 -2.7802  0.2510 -11.0743
## Dscrmn     0.7551  0.0694  10.8757
##
## Integration:
```

```
## method: Gauss-Hermite
## quadrature points: 21
##
## Optimization:
## Convergence: 0
## max(|grad|): 2.9e-05
## quasi-Newton: BFGS
```

```
anova(lsat.IRT.rasch, lsat.IRT)
```

```
##
##  Likelihood Ratio Table
##                    AIC     BIC  log.Lik  LRT df p.value
## lsat.IRT.rasch 4945.88 4975.32 -2466.94
## lsat.IRT       4953.31 5002.38 -2466.65 0.57  4   0.967
```

Note that here we can compare models using information criteria (AIC, BIC). These information criteria are only defined for ML estimation, not for (DW)LS estimation. According to AIC and BIC, we should prefer the Rasch model. Furthermore, the likelihood ratio test does not indicate a difference in fit between the 1PL and 2PL model.

Part IV: Analysis of ordered categorical items with > 2 categories

For ordered items with > 2 ordered response categories, the code is the same. Just make sure you declare the items as ordered in applying the `cfa()` function. Automatically, a threshold for the number of categories - 1 is estimated. Reverse coding is not even necessary (items that should be reverse coded just get a negative loading, but you have to make sure that all categories within an item are ordered in the same direction).

With ordered-categorical items with > 2 categoreis, you can also compare the fit of a model in which all loadings are restricted to equality (i.e., the PCM or partial credit model) with a model in which all loadings are freely estimated (i.e., the GRM or graded response model). In **`lavaan`**'s `cfa()` function, you would do this by pre-multiplying the indicators of the latent trait by the same label.

Other packages

If you want to fit the GRM and PCM using ML estimation, you can use function `grm()` from package **`ltm`**. To fit the GRM model, use function `grm()` with and specify `constrained = FALSE`. To fit the PCM, use function `gram()` and specify `constrained = TRUE`.

Alternatively, package `mirt` (short for multidimensional IRT) is probably the current state-of-the-art package when it comes to IRT analyses. As the name already suggests, it allows you to have multiple latent constructs, where IRT models traditionally assumed only a single underlying trait (or ability). Analyses using `lavaan` and `mirt` will tend to yield the same conclusions, but estimates parameters (loadings, difficulties) will differ, because the former uses least-squares estimators for ordered-categorical data, and the latter uses maximum likelihood.