Exercises ordered categorical indicator variables

Exercise 6.1

Bartholomew, Steele, Galbraith, and Moustaki (2008) analyzed four items from the British Social Attitudes Survey concerning abortion. The item responses from 379 respondents are available in the Abortion data from the ltm package. For each item, respondents were to indicate yes (1) or no (0) on whether abortion should be allowed. We will rename the items I1-I4.

```
library(ltm)
names(Abortion) <- c(paste0("I", 1:4))
```

Hint: use 'ordered = paste0("I", 1:4)' to declare the items as ordered categorical in using the cfa() function.

a) Find the proportion who endorsed each item (i.e., the mean score).

```
colMeans(Abortion)
```

```
##        I1        I2        I3        I4
## 0.4379947 0.5936675 0.6358839 0.6174142
```

b) Fit a CFA for binary responses using the CFA function, assuming a single latent variable underlies the item responses.

```
library(lavaan)
model <- '
  Theta =~ I1 + I2 + I3 + I4
'
write.table(Abortion, file = "Abortion.txt")
fit.abo <- cfa(model, data = Abortion, std.lv = TRUE,
               ordered = paste0("I", 1:4))
summary(fit.abo, standardized = TRUE)
```

```
## lavaan (0.6-1) converged normally after  13 iterations
##
##   Number of observations                           379
##
##   Estimator                                       DWLS      Robust
##   Model Fit Test Statistic                       7.291      12.647
##   Degrees of freedom                                 2           2
##   P-value (Chi-square)                           0.026       0.002
##   Scaling correction factor                                  0.587
##   Shift parameter                                            0.234
##     for simple second-order correction (Mplus variant)
##
## Parameter Estimates:
##
##   Information                                 Expected
##   Information saturated (h1) model        Unstructured
##   Standard Errors                           Robust.sem
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   Theta =~
##     I1                0.921    0.022   42.552    0.000    0.921    0.921
##     I2                0.940    0.021   44.737    0.000    0.940    0.940
##     I3                0.964    0.019   50.568    0.000    0.964    0.964
```

```
##     I4                0.905    0.025   35.507    0.000    0.905    0.905
##
## Intercepts:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .I1                0.000                               0.000    0.000
##    .I2                0.000                               0.000    0.000
##    .I3                0.000                               0.000    0.000
##    .I4                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
##     I1|t1             0.156    0.065    2.410    0.016    0.156    0.156
##     I2|t1            -0.237    0.065   -3.639    0.000   -0.237   -0.237
##     I3|t1            -0.347    0.066   -5.273    0.000   -0.347   -0.347
##     I4|t1            -0.299    0.066   -4.559    0.000   -0.299   -0.299
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .I1                0.151                               0.151    0.151
##    .I2                0.117                               0.117    0.117
##    .I3                0.071                               0.071    0.071
##    .I4                0.182                               0.182    0.182
##     Theta             1.000                               1.000    1.000
##
## Scales y*:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##     I1                1.000                               1.000    1.000
##     I2                1.000                               1.000    1.000
##     I3                1.000                               1.000    1.000
##     I4                1.000                               1.000    1.000
```

c) Evaluate overall model fit.

```
fitmeasures(fit.abo)
```

```
##                          npar                          fmin
##                         8.000                         0.010
##                         chisq                            df
##                         7.291                         2.000
##                        pvalue                  chisq.scaled
##                         0.026                        12.647
##                     df.scaled                 pvalue.scaled
##                         2.000                         0.002
##          chisq.scaling.factor                baseline.chisq
##                         0.587                      4919.479
##                   baseline.df               baseline.pvalue
##                         6.000                         0.000
##         baseline.chisq.scaled            baseline.df.scaled
##                      3905.848                         6.000
##        baseline.pvalue.scaled baseline.chisq.scaling.factor
##                         0.000                         1.260
##                           cfi                           tli
##                         0.999                         0.997
##                          nnfi                           rfi
```

```
##                        0.997                        0.996
##                          nfi                         pnfi
##                        0.999                        0.333
##                          ifi                          rni
##                        0.999                        0.999
##                   cfi.scaled                   tli.scaled
##                        0.997                        0.992
##                   cfi.robust                   tli.robust
##                           NA                           NA
##                  nnfi.scaled                  nnfi.robust
##                        0.992                           NA
##                   rfi.scaled                   nfi.scaled
##                        0.990                        0.997
##                   ifi.scaled                   rni.scaled
##                        0.997                        0.997
##                   rni.robust                        rmsea
##                           NA                        0.084
##               rmsea.ci.lower               rmsea.ci.upper
##                        0.025                        0.153
##                 rmsea.pvalue                 rmsea.scaled
##                        0.145                        0.119
##        rmsea.ci.lower.scaled        rmsea.ci.upper.scaled
##                        0.062                        0.185
##          rmsea.pvalue.scaled                 rmsea.robust
##                        0.025                           NA
##        rmsea.ci.lower.robust        rmsea.ci.upper.robust
##                           NA                           NA
##          rmsea.pvalue.robust                          rmr
##                           NA                        0.025
##                   rmr_nomean                         srmr
##                        0.029                        0.029
##                 srmr_bentler          srmr_bentler_nomean
##                        0.025                        0.029
##                  srmr_bollen           srmr_bollen_nomean
##                        0.025                        0.029
##                   srmr_mplus            srmr_mplus_nomean
##                        0.025                        0.029
##                        cn_05                        cn_01
##                      311.626                      478.508
##                          gfi                         agfi
##                        0.999                        0.993
##                         pgfi                          mfi
##                        0.200                        0.993
```

Inspect the estimated thresholds and loadings to answer the following questions:

d) If you would have to create a 1-item abortion attitude test, which item would you select?

The item with the highest discrimination parameter: Item 3.

e) If the 1-item test has to be used to find persons with extremely liberal views on abortion, which item would you select?

The item with the highest threshold (difficulty): Item 1

f) Looking at the discrimination parameters (loadings) and their standard errors, would you expect the Rasch or 2pl model to fit better?

The loadings are quite similar, they differ about 1 SE amongst each other, so the differences do not seem statistically significant. Therefore, the assumption that all loadings are equal seems tenable, and the Rasch model is probably more appropriate.

g) Statistically test whether the Rasch or 2pl model fits better.

```
model.rasch <- '
  Theta =~ 1*I1 + 1*I2 + 1*I3 + 1*I4
'
fit.rasch <- cfa(model.rasch, data = Abortion, ordered = paste0("I", 1:4))
fitinds <- c("cfi.scaled", "rmsea.scaled", "srmr")
fitMeasures(fit.abo, fitinds)
```

```
##   cfi.scaled rmsea.scaled         srmr
##        0.997        0.119        0.029
```

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

```
##   cfi.scaled rmsea.scaled         srmr
##        0.998        0.067        0.040
```

```
lavTestLRT(fit.rasch, fit.abo)
```

```
## Scaled Chi Square Difference Test (method = "satorra.2000")
##
##           Df AIC BIC  Chisq Chisq diff Df diff Pr(>Chisq)
## fit.abo    2          7.291
## fit.rasch  5         10.171     3.3525       3     0.3404
```

The chi-square difference test indicates no significant different between the fit of the two models. Then we prefer the most parsimonious model, in this case the Rasch model.

Exercise 6.2

Beaujean and Sheng (2010) conducted an IRT analysis of the ten-item vocabulary test from the General Social Survey. Data from the respondents with responses to all 10 items (n = 2943) from the 2000 decade group are available as a space delimited file (gss2000.dat), and the items are named word.a-word.j. Get the file gss2000.dat from the github repository. To load it in R, type:

```
gssdat <- read.table("gss2000.dat", header = TRUE)
```

Hint: use following code in cfa() function: ordered = paste0("word.", letters[1:3])

a) Conduct an item-level confirmatory factor analysis with one latent variable. Analyze only the first four items, as analyzing all 10 will involve a lot of typing.

```
gssmod <- '
  vocab =~ word.a + word.b + word.c + word.d
'
gssfit <- cfa(gssmod, ordered = paste0("word.", letters[1:4]), data = gssdat)
summary(gssfit, standardized = TRUE, fit.measures = TRUE)
```

```
## lavaan (0.6-1) converged normally after  23 iterations
##
##   Number of observations                          2943
##
##   Estimator                                       DWLS      Robust
##   Model Fit Test Statistic                       4.014       5.149
##   Degrees of freedom                                 2           2
```

```
##   P-value (Chi-square)                           0.134       0.076
##   Scaling correction factor                                  0.786
##   Shift parameter                                            0.044
##     for simple second-order correction (Mplus variant)
##
## Model test baseline model:
##
##   Minimum Function Test Statistic              673.679     625.493
##   Degrees of freedom                                 6           6
##   P-value                                        0.000       0.000
##
## User model versus baseline model:
##
##   Comparative Fit Index (CFI)                    0.997       0.995
##   Tucker-Lewis Index (TLI)                       0.991       0.985
##
##   Robust Comparative Fit Index (CFI)                            NA
##   Robust Tucker-Lewis Index (TLI)                               NA
##
## Root Mean Square Error of Approximation:
##
##   RMSEA                                          0.019       0.023
##   90 Percent Confidence Interval          0.000  0.045       0.000  0.049
##   P-value RMSEA <= 0.05                          0.978       0.959
##
##   Robust RMSEA                                                  NA
##   90 Percent Confidence Interval                             0.000     NA
##
## Standardized Root Mean Square Residual:
##
##   SRMR                                           0.027       0.027
##
## Parameter Estimates:
##
##   Information                                 Expected
##   Information saturated (h1) model          Unstructured
##   Standard Errors                           Robust.sem
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   vocab =~
##     word.a            1.000                               0.466    0.466
##     word.b            1.921    0.222    8.640    0.000    0.896    0.896
##     word.c            0.678    0.117    5.814    0.000    0.316    0.316
##     word.d            1.705    0.174    9.794    0.000    0.796    0.796
##
## Intercepts:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .word.a            0.000                               0.000    0.000
##    .word.b            0.000                               0.000    0.000
##    .word.c            0.000                               0.000    0.000
##    .word.d            0.000                               0.000    0.000
##     vocab             0.000                               0.000    0.000
##
```

```
## Thresholds:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##     word.a|t1         -1.061    0.029  -37.203    0.000   -1.061   -1.061
##     word.b|t1         -1.473    0.035  -42.114    0.000   -1.473   -1.473
##     word.c|t1          0.614    0.025   24.813    0.000    0.614    0.614
##     word.d|t1         -1.649    0.039  -42.209    0.000   -1.649   -1.649
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .word.a             0.782                               0.782    0.782
##    .word.b             0.197                               0.197    0.197
##    .word.c             0.900                               0.900    0.900
##    .word.d             0.367                               0.367    0.367
##     vocab              0.218    0.040    5.451    0.000    1.000    1.000
##
## Scales y*:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##     word.a             1.000                               1.000    1.000
##     word.b             1.000                               1.000    1.000
##     word.c             1.000                               1.000    1.000
##     word.d             1.000                               1.000    1.000
```

Model fit is perfect (by definition, there are only three indicators), standardized loadings are substantial.

b) What are the easiest and most difficult items?

Easiest item is word.d, most difficult item is word.c.

c) What are the best and worst indicators of the latent trait?

Best indicator is word.b, worst indicator is word.c.

d) Does the Rasch, or the 2pl model fit the 3 vocabulary items better?

```r
gssmod.rasch <- '
  vocab =~ a*word.a + a*word.b + a*word.c + a*word.d
'
gssfit.rasch <- cfa(gssmod.rasch, ordered = paste0("word.", letters[1:4]),
                    data = gssdat)
fitMeasures(gssfit, fitinds)
```

```
##   cfi.scaled rmsea.scaled         srmr
##        0.995        0.023        0.027
```

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

```
##   cfi.scaled rmsea.scaled         srmr
##        0.767        0.099        0.142
```

```r
lavTestLRT(gssfit.rasch, gssfit)
```

```
## Scaled Chi Square Difference Test (method = "satorra.2000")
##
##              Df AIC BIC    Chisq Chisq diff Df diff Pr(>Chisq)
## gssfit        2           4.0139
## gssfit.rasch  5         137.0874     135.42       3  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The fit of the 2pl model is significantly better than that of the Rasch model. So for this vocabulary test, maybe the 2pl model should be preferred. However, the sample size is very large, so we have a lot of power to detect differences which may in fact be small.

Additional exercise: HADS

```
library(foreign)
HADS <- read.spss("HADS.sav", use.value.labels = TRUE, to.data.frame = TRUE)
summary(HADS)
```

```
##  Respondentnummer    leeftijd           geslacht             HADS1
##  Min.   :500002   Min.   :18.00   een man   :217   bijna nooit : 43
##  1st Qu.:500162   1st Qu.:35.00   een vrouw :285   soms        :160
##  Median :500333   Median :43.00                    vaak        :202
##  Mean   :500335   Mean   :42.84                    bijna altijd: 97
##  3rd Qu.:500512   3rd Qu.:51.00
##  Max.   :500689   Max.   :80.00
##          HADS2             HADS3             HADS4
##  bijna nooit :214   bijna nooit : 75   bijna altijd: 31
##  soms        :151   soms        :175   vaak        : 81
##  vaak        :103   vaak        :180   soms        :219
##  bijna altijd: 34   bijna altijd: 72   bijna nooit :171
##
##
##          HADS5             HADS6             HADS7
##  bijna nooit :179   bijna nooit : 67   bijna nooit :199
##  soms        :170   soms        :204   soms        :187
##  vaak        :116   vaak        :167   vaak        :101
##  bijna altijd: 37   bijna altijd: 64   bijna altijd: 15
##
##
```

a) Fit a graded response model to the data:

```
HADS.GRM.mod <- '
  anx =~ HADS1 + HADS2 + HADS3 + HADS4 + HADS5 + HADS6 + HADS7
'
HADS.GRM.fit <- cfa(HADS.GRM.mod, data = HADS, ordered = paste("HADS", 1:7, sep=""))
```

The warning about empty bivariate cell tables is common, do not worry about this. The bivariate tables are frequency tables, with response categories of one item in the rows and those of another item in the columns. These bivariate tables are used to calculate the tetrachoric correlation matrix. With 7 items with 4 response categories, there are $7 \times (7-1)$ bivariate tables, with 4×4 cells in each table. If twelve of them are empty, that is certainly not a lot.

```
summary(HADS.GRM.fit, standardized = TRUE)
```

```
## lavaan (0.6-1) converged normally after  18 iterations
##
##   Number of observations                           502
##
##   Estimator                                       DWLS      Robust
##   Model Fit Test Statistic                      94.652     171.090
##   Degrees of freedom                                14          14
##   P-value (Chi-square)                           0.000       0.000
```

```
##   Scaling correction factor                                  0.559
##   Shift parameter                                            1.733
##     for simple second-order correction (Mplus variant)
##
## Parameter Estimates:
##
##   Information                                 Expected
##   Information saturated (h1) model        Unstructured
##   Standard Errors                           Robust.sem
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   anx =~
##     HADS1             1.000                               0.832    0.832
##     HADS2             0.961    0.033   29.081    0.000    0.799    0.799
##     HADS3             0.962    0.033   29.428    0.000    0.800    0.800
##     HADS4             0.756    0.042   18.089    0.000    0.629    0.629
##     HADS5             0.737    0.041   18.153    0.000    0.613    0.613
##     HADS6             0.878    0.034   25.691    0.000    0.730    0.730
##     HADS7             0.912    0.034   27.160    0.000    0.759    0.759
##
## Intercepts:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .HADS1             0.000                               0.000    0.000
##    .HADS2             0.000                               0.000    0.000
##    .HADS3             0.000                               0.000    0.000
##    .HADS4             0.000                               0.000    0.000
##    .HADS5             0.000                               0.000    0.000
##    .HADS6             0.000                               0.000    0.000
##    .HADS7             0.000                               0.000    0.000
##     anx               0.000                               0.000    0.000
##
## Thresholds:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##     HADS1|t1         -1.368    0.080  -17.124    0.000   -1.368   -1.368
##     HADS1|t2         -0.242    0.057   -4.276    0.000   -0.242   -0.242
##     HADS1|t3          0.866    0.064   13.462    0.000    0.866    0.866
##     HADS2|t1         -0.186    0.056   -3.298    0.001   -0.186   -0.186
##     HADS2|t2          0.604    0.060   10.089    0.000    0.604    0.604
##     HADS2|t3          1.493    0.086   17.407    0.000    1.493    1.493
##     HADS3|t1         -1.039    0.068  -15.170    0.000   -1.039   -1.039
##     HADS3|t2         -0.005    0.056   -0.089    0.929   -0.005   -0.005
##     HADS3|t3          1.065    0.069   15.388    0.000    1.065    1.065
##     HADS4|t1         -1.540    0.088  -17.450    0.000   -1.540   -1.540
##     HADS4|t2         -0.762    0.062  -12.224    0.000   -0.762   -0.762
##     HADS4|t3          0.411    0.058    7.113    0.000    0.411    0.411
##     HADS5|t1         -0.368    0.057   -6.406    0.000   -0.368   -0.368
##     HADS5|t2          0.511    0.059    8.696    0.000    0.511    0.511
##     HADS5|t3          1.449    0.084   17.336    0.000    1.449    1.449
##     HADS6|t1         -1.110    0.071  -15.740    0.000   -1.110   -1.110
##     HADS6|t2          0.100    0.056    1.783    0.075    0.100    0.100
##     HADS6|t3          1.138    0.071   15.944    0.000    1.138    1.138
##     HADS7|t1         -0.263    0.057   -4.632    0.000   -0.263   -0.263
##     HADS7|t2          0.735    0.062   11.887    0.000    0.735    0.735
```

```
##     HADS7|t3          1.883    0.112   16.784    0.000    1.883    1.883
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .HADS1             0.308                               0.308    0.308
##    .HADS2             0.361                               0.361    0.361
##    .HADS3             0.360                               0.360    0.360
##    .HADS4             0.604                               0.604    0.604
##    .HADS5             0.624                               0.624    0.624
##    .HADS6             0.467                               0.467    0.467
##    .HADS7             0.424                               0.424    0.424
##     anx               0.692    0.033   21.088    0.000    1.000    1.000
##
## Scales y*:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##     HADS1             1.000                               1.000    1.000
##     HADS2             1.000                               1.000    1.000
##     HADS3             1.000                               1.000    1.000
##     HADS4             1.000                               1.000    1.000
##     HADS5             1.000                               1.000    1.000
##     HADS6             1.000                               1.000    1.000
##     HADS7             1.000                               1.000    1.000
```

```r
fitMeasures(HADS.GRM.fit, fitinds)
```

```
##   cfi.scaled rmsea.scaled         srmr
##        0.954        0.150        0.066
```

b) Which category from which item is the ‘easiest’?

HADS4, it has the lowest thresholds for all categories. Note that you can also see this from the histograms printed earlier.

c) What do we mean by ‘easiest’ in this case?

For for this item, lower latent trait (Anxiety) values are needed to endorse higher response categories.

d) Are all category thresholds ordered similarly across items?

Yes, they go from low to high.

e) Fit a partial credit model to the data:

```r
HADS.PCM.mod <- '
  anx =~ 1*HADS1 + 1*HADS2 + 1*HADS3 + 1*HADS4 + 1*HADS5 + 1*HADS6 + 1*HADS7
'
HADS.PCM.fit <- cfa(HADS.PCM.mod, data = HADS, ordered = paste("HADS", 1:7, sep=""))
summary(HADS.PCM.fit, standardized = TRUE)
```

```
## lavaan (0.6-1) converged normally after   3 iterations
##
##   Number of observations                           502
##
##   Estimator                                       DWLS      Robust
##   Model Fit Test Statistic                     192.056     206.433
##   Degrees of freedom                                20          20
##   P-value (Chi-square)                           0.000       0.000
##   Scaling correction factor                                  0.950
##   Shift parameter                                            4.277
```

```
##     for simple second-order correction (Mplus variant)
##
## Parameter Estimates:
##
##   Information                                 Expected
##   Information saturated (h1) model          Unstructured
##   Standard Errors                           Robust.sem
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   anx =~
##     HADS1      (l)    1.000                                0.750    0.750
##     HADS2      (l)    1.000                                0.750    0.750
##     HADS3      (l)    1.000                                0.750    0.750
##     HADS4      (l)    1.000                                0.750    0.750
##     HADS5      (l)    1.000                                0.750    0.750
##     HADS6      (l)    1.000                                0.750    0.750
##     HADS7      (l)    1.000                                0.750    0.750
##
## Intercepts:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .HADS1             0.000                                0.000    0.000
##    .HADS2             0.000                                0.000    0.000
##    .HADS3             0.000                                0.000    0.000
##    .HADS4             0.000                                0.000    0.000
##    .HADS5             0.000                                0.000    0.000
##    .HADS6             0.000                                0.000    0.000
##    .HADS7             0.000                                0.000    0.000
##     anx               0.000                                0.000    0.000
##
## Thresholds:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##     HADS1|t1         -1.368    0.080  -17.124    0.000   -1.368   -1.368
##     HADS1|t2         -0.242    0.057   -4.276    0.000   -0.242   -0.242
##     HADS1|t3          0.866    0.064   13.462    0.000    0.866    0.866
##     HADS2|t1         -0.186    0.056   -3.298    0.001   -0.186   -0.186
##     HADS2|t2          0.604    0.060   10.089    0.000    0.604    0.604
##     HADS2|t3          1.493    0.086   17.407    0.000    1.493    1.493
##     HADS3|t1         -1.039    0.068  -15.170    0.000   -1.039   -1.039
##     HADS3|t2         -0.005    0.056   -0.089    0.929   -0.005   -0.005
##     HADS3|t3          1.065    0.069   15.388    0.000    1.065    1.065
##     HADS4|t1         -1.540    0.088  -17.450    0.000   -1.540   -1.540
##     HADS4|t2         -0.762    0.062  -12.224    0.000   -0.762   -0.762
##     HADS4|t3          0.411    0.058    7.113    0.000    0.411    0.411
##     HADS5|t1         -0.368    0.057   -6.406    0.000   -0.368   -0.368
##     HADS5|t2          0.511    0.059    8.696    0.000    0.511    0.511
##     HADS5|t3          1.449    0.084   17.336    0.000    1.449    1.449
##     HADS6|t1         -1.110    0.071  -15.740    0.000   -1.110   -1.110
##     HADS6|t2          0.100    0.056    1.783    0.075    0.100    0.100
##     HADS6|t3          1.138    0.071   15.944    0.000    1.138    1.138
##     HADS7|t1         -0.263    0.057   -4.632    0.000   -0.263   -0.263
##     HADS7|t2          0.735    0.062   11.887    0.000    0.735    0.735
##     HADS7|t3          1.883    0.112   16.784    0.000    1.883    1.883
##
```

```
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .HADS1             0.438                               0.438    0.438
##    .HADS2             0.438                               0.438    0.438
##    .HADS3             0.438                               0.438    0.438
##    .HADS4             0.438                               0.438    0.438
##    .HADS5             0.438                               0.438    0.438
##    .HADS6             0.438                               0.438    0.438
##    .HADS7             0.438                               0.438    0.438
##     anx               0.562    0.019   30.186    0.000    1.000    1.000
##
## Scales y*:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##     HADS1             1.000                               1.000    1.000
##     HADS2             1.000                               1.000    1.000
##     HADS3             1.000                               1.000    1.000
##     HADS4             1.000                               1.000    1.000
##     HADS5             1.000                               1.000    1.000
##     HADS6             1.000                               1.000    1.000
##     HADS7             1.000                               1.000    1.000
```

Note that again we see Item 4 is the easiest item, with the lowest thresholds.

f) Test whether the GRM or PCM fits better:

```
fitMeasures(HADS.GRM.fit, fitinds)
```

```
##   cfi.scaled rmsea.scaled         srmr
##        0.954        0.150        0.066
```

```
fitMeasures(HADS.PCM.fit, fitinds)
```

```
##   cfi.scaled rmsea.scaled         srmr
##        0.946        0.136        0.097
```

```
anova(HADS.PCM.fit, HADS.GRM.fit)
```

```
## Scaled Chi Square Difference Test (method = "satorra.2000")
##
##              Df AIC BIC   Chisq Chisq diff Df diff Pr(>Chisq)
## HADS.GRM.fit 14          94.652
## HADS.PCM.fit 20         192.056     67.696       6  1.213e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

There is a significant difference in fit, so we should prefer the GRM, which is more complex. This is also indicated by the CFI values. However, if we use the RMSEA as the main criterion for model selection, we would prefer the PCM, because it is more parsimonious.