Answers to exercises ordered categorical indicator variables

Additional Exercise: HADS

```r
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             HADS5
##  bijna nooit :214   bijna nooit : 75   bijna altijd: 31   bijna nooit :179
##  soms        :151   soms        :175   vaak        : 81   soms        :170
##  vaak        :103   vaak        :180   soms        :219   vaak        :116
##  bijna altijd: 34   bijna altijd: 72   bijna nooit :171   bijna altijd: 37
##
##
##           HADS6             HADS7
##  bijna nooit : 67   bijna nooit :199
##  soms        :204   soms        :187
##  vaak        :167   vaak        :101
##  bijna altijd: 64   bijna altijd: 15
##
##
```

a) To fit a graded response model to the data:

```r
library("lavaan")
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=""))
summary(HADS.GRM.fit, standardized = TRUE)
```

```
## lavaan 0.6-5 ended normally after 18 iterations
##
##   Estimator                                       DWLS
##   Optimization method                           NLMINB
##   Number of free parameters                         28
##
##   Number of observations                           502
##
## Model Test User Model:
```

```
##                                               Standard      Robust
##   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 the simple second-order correction
##
## 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
```

b) HADS4 seems to be the easiest item, because it has the lowest thresholds for all categories.

c) With 'easiest', we mean that for this item, lower latent trait (anxiety) levels are needed to endorse a higher response category.

d) Yes, all category thresholds are ordered similarly across items; they go from low to high.

e) To fit a partial credit model to the data, we pre-multiple the indicators by the same label:

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

```
## lavaan 0.6-5 ended normally after 3 iterations
##
##   Estimator                                       DWLS
##   Optimization method                           NLMINB
##   Number of free parameters                         22
##
##   Number of observations                           502
##
## Model Test User Model:
##                                               Standard      Robust
##   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 the simple second-order correction
##
## 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) The standardized loadings in the GRM differ only somewhat between items, with the largest difference around .2. So we could prefer the PCM for that reason. But if we want to be able to distinguish between items that discrimintate more or less well, we could prefer the GRM.

Let's test the difference in fit and inspect model fit indiceS:

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

```
##  chisq.scaled            df pvalue.scaled    cfi.scaled  rmsea.scaled
##       171.090        14.000         0.000         0.954         0.150
##          srmr
##         0.066
```

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

```
##  chisq.scaled            df pvalue.scaled    cfi.scaled  rmsea.scaled
##       206.433        20.000         0.000         0.946         0.136
##          srmr
##         0.097
```

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

```
## 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)
## 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 from a statistical point of view 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.

Would we reach the same conclusion using ML estimation?

```
library("ltm")
GRM.IRT <- grm(HADS[ , 4:10])
coef(GRM.IRT)
```

```
##       Extrmt1 Extrmt2 Extrmt3 Dscrmn
## HADS1  -1.668  -0.269   1.025  2.610
## HADS2  -0.236   0.718   1.863  2.264
## HADS3  -1.254   0.002   1.268  2.533
## HADS4  -2.540  -1.168   0.655  1.365
## HADS5  -0.598   0.786   2.335  1.338
## HADS6  -1.511   0.154   1.541  1.870
## HADS7  -0.332   0.923   2.485  2.062
```

```
PCM.IRT <- grm(HADS[ , 4:10], constrained = TRUE)
anova(PCM.IRT, GRM.IRT)
```

```
##
##  Likelihood Ratio Table
##             AIC     BIC  log.Lik   LRT df p.value
## PCM.IRT 7579.43 7672.24 -3767.72
## GRM.IRT 7532.45 7650.57 -3738.22 58.99  6  <0.001
```

The ML-estimated GRM indicates highest discriminatory power for HADS item 1, followed by HADS items 3, 2 and 7. This is similar to what we found using DWLS estimation. Also, item 4 seems most easy, both with ML and DWLS estimation.

The likelihood ratio test, AIC and BIC all indicate that the GRM fits the data better than the PCM.

g)

First, we convert the HADS items to numeric:

```
HADS2 <- sapply(HADS[ , 4:10], as.numeric)
```

Then we fit a CFA to the numeric items. We can use the same model specification as for the GRM, and have to specify the type of estimator used:

```
HADS.ML.fit <- cfa(HADS.GRM.mod, data = HADS2, meanstructure = TRUE)
parameterestimates(HADS.ML.fit, standardized = TRUE)[ , c(1:5, 7, 11)]
```

```
##      lhs op   rhs   est    se pvalue std.all
## 1    anx =~ HADS1 1.000 0.000     NA   0.772
## 2    anx =~ HADS2 0.982 0.064      0   0.702
## 3    anx =~ HADS3 1.029 0.062      0   0.761
## 4    anx =~ HADS4 0.713 0.059      0   0.558
## 5    anx =~ HADS5 0.774 0.065      0   0.558
## 6    anx =~ HADS6 0.865 0.060      0   0.666
## 7    anx =~ HADS7 0.831 0.057      0   0.672
## 8  HADS1 ~~ HADS1 0.309 0.026      0   0.403
## 9  HADS2 ~~ HADS2 0.454 0.034      0   0.507
## 10 HADS3 ~~ HADS3 0.351 0.028      0   0.420
## 11 HADS4 ~~ HADS4 0.514 0.035      0   0.689
```

```
## 12 HADS5 ~~ HADS5 0.608 0.041      0   0.689
## 13 HADS6 ~~ HADS6 0.428 0.031      0   0.556
## 14 HADS7 ~~ HADS7 0.383 0.028      0   0.548
## 15   anx ~~   anx 0.457 0.047      0   1.000
## 16 HADS1 ~1        2.703 0.039      0   3.088
## 17 HADS2 ~1        1.914 0.042      0   2.023
## 18 HADS3 ~1        2.496 0.041      0   2.730
## 19 HADS4 ~1        3.056 0.039      0   3.538
## 20 HADS5 ~1        2.022 0.042      0   2.153
## 21 HADS6 ~1        2.454 0.039      0   2.797
## 22 HADS7 ~1        1.865 0.037      0   2.230
## 23   anx ~1        0.000 0.000     NA   0.000
```

The standardized loadings indicate that item 1 is the best indicator, followed by item 3, 2 and then 7. So in that respect, treating the items a continuous or ordered does not really seem to make a difference.

The item intercepts indicate that item 4 is the easiest item. Item intercepts are the expected value of the item score, when the LV has a value of 0. So, the higher the item intercept, the higher the item score given the same latent trait value.

The standardized loadings are a bit lower in the model where we treat the indicators as continuous. The residual variance are higher in the model where we treat the indicators as continuous. This is in line with the very first observation we made in Example 6.2: Pearson correlations (assuming continuous variables) are lower than tetra- and polychoric correlations (which assume ordered categorical variables, which arise from an underlying continuous latent variable).

```r
fitinds2 <- c("chisq", "df", "pvalue", "cfi", "rmsea", "srmr")
fitmeasures(HADS.ML.fit, fitinds2)
```

```
##   chisq      df  pvalue     cfi   rmsea    srmr
## 149.170  14.000   0.000   0.898   0.139   0.053
```

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

```
##  chisq.scaled            df pvalue.scaled    cfi.scaled  rmsea.scaled
##       171.090        14.000         0.000         0.954         0.150
##          srmr
##         0.066
```

The model fit does differ quite a bit between the models, which is to be expected.

In conclusion: Treating ordered-categorical items as continuous may not be accurate, but it will likely give you similar results as fitting an ordered-categorical item factor analysis.

When ordered-categorical items can be treated as continuous has been rigourously studies by several authors (see references below). Rhemtulla et al. (2012) recommend treating item responses as continuous only when they have at least 5 ordered categories.

Dolan, C. V. (1994). Factor analysis of variables with 2, 3, 5 and 7 response categories: A comparison of categorical variable estimators using simulated data. *British Journal of Mathematical and Statistical Psychology, 47*(2), 309-326.

DiStefano, C. (2002). The impact of categorization with confirmatory factor analysis. *Structural Equation Modeling, 9*(3), 327-346.

Rhemtulla, M., Brosseau-Liard, P. E., & Savalei, V. (2012). When can categorical variables be treated as continuous? A comparison of robust continuous and categorical SEM estimation methods under suboptimal conditions. *Psychological Methods, 17*(3), 354.

h) Now we use robust ML:

```r
HADS.MLR.fit <- cfa(HADS.GRM.mod, data = HADS2, estimator = "MLR", meanstructure = TRUE)
parameterestimates(HADS.MLR.fit, standardized = TRUE)[ , c(1:5, 7, 11)]
```

```
##      lhs op   rhs   est    se pvalue std.all
## 1    anx =~ HADS1 1.000 0.000     NA   0.772
## 2    anx =~ HADS2 0.982 0.079      0   0.702
## 3    anx =~ HADS3 1.029 0.065      0   0.761
## 4    anx =~ HADS4 0.713 0.058      0   0.558
## 5    anx =~ HADS5 0.774 0.065      0   0.558
## 6    anx =~ HADS6 0.865 0.050      0   0.666
## 7    anx =~ HADS7 0.831 0.067      0   0.672
## 8  HADS1 ~~ HADS1 0.309 0.031      0   0.403
## 9  HADS2 ~~ HADS2 0.454 0.039      0   0.507
## 10 HADS3 ~~ HADS3 0.351 0.035      0   0.420
## 11 HADS4 ~~ HADS4 0.514 0.037      0   0.689
## 12 HADS5 ~~ HADS5 0.608 0.040      0   0.689
## 13 HADS6 ~~ HADS6 0.428 0.039      0   0.556
## 14 HADS7 ~~ HADS7 0.383 0.030      0   0.548
## 15   anx ~~   anx 0.457 0.045      0   1.000
## 16 HADS1 ~1       2.703 0.039      0   3.088
## 17 HADS2 ~1       1.914 0.042      0   2.023
## 18 HADS3 ~1       2.496 0.041      0   2.730
## 19 HADS4 ~1       3.056 0.039      0   3.538
## 20 HADS5 ~1       2.022 0.042      0   2.153
## 21 HADS6 ~1       2.454 0.039      0   2.797
## 22 HADS7 ~1       1.865 0.037      0   2.230
## 23   anx ~1       0.000 0.000     NA   0.000
```

We get identical parameter estimates as with standard ML.

```r
fitinds2 <- c("chisq.scaled", "df", "pvalue.scaled", "cfi.robust",
              "rmsea.robust", "srmr")
fitmeasures(HADS.MLR.fit, fitinds2)
```

```
##  chisq.scaled            df pvalue.scaled    cfi.robust  rmsea.robust 
##       131.093        14.000         0.000         0.899         0.138 
##          srmr 
##         0.053
```

The robust fit indices indicate slightly better fit, but the difference with standard ML seems small.