Ilustración análisis de invarianza en R con lavaan y semTools

Dr. Héctor Nájera y Dr. Curtis Huffman

12 May 2020

Introducción

Esta nota ilustra cómo se puede hacer un análisis de invarianza de grupos múltiples (MGFCA, en inglés) con los paquetes `lavaan()` y `semTools()`. Vamos a utilizar el famoso ejemplo de 1939 de Holtzinger y cuyos datos están precargados en el paquete `lavaan()`.

```
library(lavaan)
library(semTools)
```

Con `semTools` el análisis es más fácil pero no es la mejor alternativa. Veremos la ventaja de `lavaan()` en términos de los ajustes que hay que hacer "a mano" al modelo. Aquí es donde es relevante la lectura sobre los grados de libertad del investigador.

Pasos análisis de invarianza

Como recordatorio, se tienen los siguientes pasos:

1. Paso 1: Especificación del modelo factorial
2. Paso 2: Estimación del modelo factorial, considerando ambos grupos, con todos los parámetros "libres" entre grupos (i.e. que el modelo estime lo que tenga que estimar y encuentre el mejor ajuste posible)
3. Paso 3: Checar los valores globales de ajuste. Si los indicadores de TLI, CFI, BIC, AIC y Chi-2 son aceptables con los criterios que hemos utilizado, entonces podemos seguir. Si no... NO! El modelo no se sostiene.
4. Paso 4: Estimar el modelo de invarianza métrica (igualdad de cargas factoriales). Ese decir, imponemos la restricción de que las cargas factoriales son iguales entre grupos.
5. Paso 5: Checar el ajuste del modelo y compararlo con el anterior. Si no hay cambios, entonces seguimos. Si el modelo empeora quiere decir que la varianza métrica no se sostiene.
6. Paso 6: Si el modelo escalar se sostiene, entonces pasamos a estimar el modelo de invarianza métrica. Es decir, el modelo con las constantes fijas (iguales) entre grupos.
7. Paso 7: Checamos el ajuste del modelo en los términos arriba descritos.

Análisis de invarianza (fácil y rápido)

Usaremos el paquete `semTools` primero. Veremos que es muy práctico (porque automatiza el análisis) pero no necesariamente el mejor porque hay problemas.

Paso 1: El primer paso es la especificación del modelo factorial que propone tres dimensiones en el ámbito del aprendizaje: visual, textual y velocidad. Esto se hace a continuación:

```
ModeloHS <-'visual =~ x1 + x2 + x3
            textual =~ x4 + x5 + x6
            speed =~ x7 + x8 + x9'
```

Paso 2: El segundo paso es estimar una serie de modelos para checar los distintos niveles de invarianza. Como recordatorio se estiman del más flexible al más rígido:

1. Configural
2. Métrica
3. Escalar

El paquete `semTools` tiene la ventaja de estimar estos modelos de manera automática. Los requerimientos mínimos son: El nombre del modelo, los datos y los grupos (i.e. los grupos para los que vamos a comparar si invarianza se sostiene)

Los datos se encuentran en el objeto `HolzingerSwineford1939`.

```
library(semTools)
Invarianza<-measurementInvariance(model = ModeloHS,
                                  data = HolzingerSwineford1939,group = "school",
                                  quiet = TRUE)
```

Ahora exploramos el objeto `Invarianza`:

```
compareFit(Invarianza)
```

```
## ################### Nested Model Comparison #########################
## Chi-Squared Difference Test
##
##                Df    AIC    BIC  Chisq Chisq diff Df diff Pr(>Chisq)
## fit.configural 48 7484.4 7706.8 115.85
## fit.loadings   54 7480.6 7680.8 124.04      8.192       6     0.2244
## fit.intercepts 60 7508.6 7686.6 164.10     40.059       6  4.435e-07 ***
## fit.means      63 7543.1 7710.0 204.61     40.502       3  8.338e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ######################## Model Fit Indices ###########################
##                   chisq df pvalue   cfi   tli       aic       bic rmsea
## fit.configural 115.851† 48   .000 .923† .885  7484.395  7706.822  .097
## fit.loadings   124.044  54   .000 .921  .895† 7480.587† 7680.771† .093†
## fit.intercepts 164.103  60   .000 .882  .859  7508.647  7686.588  .107
## fit.means      204.605  63   .000 .840  .817  7543.149  7709.969  .122
##                 srmr
## fit.configural .068†
## fit.loadings   .072
## fit.intercepts .082
## fit.means      .109
##
## ################### Differences in Fit Indices ######################
##                               df    cfi    tli    aic     bic  rmsea  srmr
## fit.loadings - fit.configural  6 -0.002  0.009 -3.808 -26.050 -0.004 0.004
## fit.intercepts - fit.loadings  6 -0.038 -0.036 28.059   5.817  0.015 0.011
## fit.means - fit.means          3 -0.042 -0.042 34.502  23.381  0.015 0.026
```

Qué concluímos? Bueno que la invarianza configural y la métrica se sostienen pero la escalar no!!! Por qué? Vean los valores de ajuste global y los cambios de un modelo a otro. Lo que queremos es que todos los modelos sean igual de buenos. Entonces, qué podemos hacer?

Ahora con lavaan

Invarianza configural

Estimamos el modelo `ModeloHS` con `lavaan()` con la opción `group`. Después pedimos los estadísticos de ajuste del modelo.

```
model1<- cfa(ModeloHS, data=HolzingerSwineford1939, group="school")
fitmeasures(model1)
```

```
##                 npar                 fmin                chisq
##               60.000                0.192              115.851
##                   df               pvalue       baseline.chisq
##               48.000                0.000              957.769
##          baseline.df     baseline.pvalue                  cfi
##               72.000                0.000                0.923
##                  tli                 nnfi                  rfi
##                0.885                0.885                0.819
##                  nfi                 pnfi                  ifi
##                0.879                0.586                0.925
##                  rni                 logl    unrestricted.logl
##                0.923            -3682.198            -3624.272
##                  aic                  bic               ntotal
##             7484.395             7706.822              301.000
##                 bic2                rmsea       rmsea.ci.lower
##             7516.536                0.097                0.075
##       rmsea.ci.upper        rmsea.pvalue                  rmr
##                0.120                0.001                0.083
##           rmr_nomean                 srmr        srmr_bentler
##                0.091                0.068                0.068
## srmr_bentler_nomean                 crmr         crmr_nomean
##                0.074                0.074                0.083
##           srmr_mplus    srmr_mplus_nomean                cn_05
##                0.068                0.074              170.324
##                cn_01                  gfi                 agfi
##              192.439                0.995                0.989
##                 pgfi                  mfi
##                0.442                0.893
```

Ahora vamos a dividir los parámetros por grupo para que se más fácil su inspección:

```
grupo1<-parameterEstimates(model1)[1:36,]
grupo2<-parameterEstimates(model1)[37:72,]
```

Cargas factoriales:

```
cbind(grupo1[1:9,c(1,2,3,5,6)],grupo2[1:9,c(5,6)])
```

```
##       lhs op rhs group       est group       est
## 1  visual =~  x1     1 1.0000000     2 1.0000000
## 2  visual =~  x2     1 0.3937180     2 0.7361616
## 3  visual =~  x3     1 0.5699292     2 0.9247992
```

```
## 4 textual =~  x4     1 1.0000000     2 1.0000000
## 5 textual =~  x5     1 1.1833321     2 0.9897913
## 6 textual =~  x6     1 0.8749601     2 0.9633399
## 7   speed =~  x7     1 1.0000000     2 1.0000000
## 8   speed =~  x8     1 1.1246900     2 1.2258375
## 9   speed =~  x9     1 0.9220018     2 1.0579038
```

Ordenada al origen (constante):

```
cbind(grupo1[25:33,c(1,2,3,5,6)],grupo2[25:33,c(5,6)])
```

```
##    lhs op rhs group      est group      est
## 25  x1 ~1         1 4.941239     2 4.929885
## 26  x2 ~1         1 5.983974     2 6.200000
## 27  x3 ~1         1 2.487179     2 1.995690
## 28  x4 ~1         1 2.822650     2 3.317241
## 29  x5 ~1         1 3.995192     2 4.712069
## 30  x6 ~1         1 1.922161     2 2.468966
## 31  x7 ~1         1 4.432274     2 3.920840
## 32  x8 ~1         1 5.563141     2 5.488276
## 33  x9 ~1         1 5.417735     2 5.327203
```

Invarianza métrica

Ahora estimamos el modelo para examinar si la invarinza métrica se sostiene. Esto se hace con la opción `group.equal=c("loadings")`. Observamos pocos cambios en el ajuste global.

```
model1.m<- cfa(ModeloHS, data=HolzingerSwineford1939,
               group="school", group.equal=c("loadings"))
fitmeasures(model1.m)
```

```
##                 npar                 fmin                chisq
##               54.000                0.206              124.044
##                   df               pvalue       baseline.chisq
##               54.000                0.000              957.769
##          baseline.df     baseline.pvalue                  cfi
##               72.000                0.000                0.921
##                  tli                 nnfi                  rfi
##                0.895                0.895                0.827
##                  nfi                 pnfi                  ifi
##                0.870                0.653                0.922
##                  rni                 logl    unrestricted.logl
##                0.921            -3686.294            -3624.272
##                  aic                  bic               ntotal
##             7480.587             7680.771              301.000
##                 bic2                rmsea       rmsea.ci.lower
##             7509.514                0.093                0.071
##       rmsea.ci.upper        rmsea.pvalue                  rmr
##                0.114                0.001                0.089
##           rmr_nomean                 srmr        srmr_bentler
##                0.098                0.072                0.072
## srmr_bentler_nomean                 crmr         crmr_nomean
```

```
##               0.078               0.081               0.085
##          srmr_mplus   srmr_mplus_nomean               cn_05
##               0.075               0.078             176.085
##               cn_01                 gfi                agfi
##             197.719               0.995               0.989
##                pgfi                 mfi
##               0.497               0.890
```

Otra forma es checar con un test de diferencias de Chi cuadrada. Vemos que no hay diferencias:

```
anova(model1,model1.m)
```

```
## Chi-Squared Difference Test
##
##          Df    AIC    BIC  Chisq Chisq diff Df diff Pr(>Chisq)
## model1   48 7484.4 7706.8 115.85
## model1.m 54 7480.6 7680.8 124.04     8.1922       6     0.2244
```

Invarianza escalar

Ahora vamos a hacer la estimación pero con contantes fijas también:

```
model1.e<- cfa(ModeloHS, data=HolzingerSwineford1939, group="school",
               group.equal=c("loadings","intercepts"))
fitmeasures(model1.e)
```

```
##                npar                fmin               chisq
##              48.000               0.273             164.103
##                  df              pvalue      baseline.chisq
##              60.000               0.000             957.769
##         baseline.df     baseline.pvalue                 cfi
##              72.000               0.000               0.882
##                 tli                nnfi                 rfi
##               0.859               0.859               0.794
##                 nfi                pnfi                 ifi
##               0.829               0.691               0.884
##                 rni                logl   unrestricted.logl
##               0.882           -3706.323           -3624.272
##                 aic                 bic              ntotal
##            7508.647            7686.588             301.000
##                bic2               rmsea      rmsea.ci.lower
##            7534.359               0.107               0.088
##      rmsea.ci.upper        rmsea.pvalue                 rmr
##               0.127               0.000               0.100
##          rmr_nomean                srmr        srmr_bentler
##               0.099               0.082               0.082
## srmr_bentler_nomean                crmr         crmr_nomean
##               0.080               0.093               0.084
##          srmr_mplus   srmr_mplus_nomean               cn_05
##               0.087               0.078             146.053
##               cn_01                 gfi                agfi
```

```
##             163.107               0.993               0.987
##                pgfi                 mfi
##               0.552               0.841
```

Podemos ahora hacer el test de Chi cuadrada. Vemos que el cambio en el ajuste es significativo y por tanto la invarianza escalar no se sostiene.

```
anova(model1,model1.e)
```

```
## Chi-Squared Difference Test
##
##          Df    AIC    BIC  Chisq Chisq diff Df diff Pr(>Chisq)
## model1   48 7484.4 7706.8 115.85
## model1.e 60 7508.6 7686.6 164.10     48.251      12  2.826e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Cómo podemos mejorar el modelo? Quizá no todos las constantes son invariantes pero algunos lo serán. Los ítems x3 y x7 tienen los mayores problemas. Esto lo sabemos a través de los índices de modificación o el test de parámetros de `lavaan`. Esto se hace con la función `lavTestScore()`.

Primero checamos la contribución de cada indicador al **desajuste** (entre más alto el número **X2** mayor el desajuste). Vemos en la columna lhs que p12 y p 16 tienen la mayor contribución. Qué significan las **p's**? Por alguna razón (no vemos ninguna buena) se trata de las etiquetas de los parámetros estimados. Necesitamos checar las etiquetas:

```
lavTestScore(model1.e)
```

```
## $test
##
## total score test:
##
##    test     X2 df p.value
## 1 score 46.956 15       0
##
## $uni
##
## univariate score tests:
##
##      lhs op   rhs     X2 df p.value
## 1   .p2. == .p38.  0.306  1   0.580
## 2   .p3. == .p39.  1.636  1   0.201
## 3   .p5. == .p41.  2.744  1   0.098
## 4   .p6. == .p42.  2.627  1   0.105
## 5   .p8. == .p44.  0.027  1   0.871
## 6   .p9. == .p45.  0.004  1   0.952
## 7  .p25. == .p61.  5.847  1   0.016
## 8  .p26. == .p62.  6.863  1   0.009
## 9  .p27. == .p63. 19.193  1   0.000
## 10 .p28. == .p64.  2.139  1   0.144
## 11 .p29. == .p65.  1.563  1   0.211
## 12 .p30. == .p66.  0.032  1   0.857
## 13 .p31. == .p67. 15.021  1   0.000
## 14 .p32. == .p68.  4.710  1   0.030
## 15 .p33. == .p69.  1.498  1   0.221
```

Para checar las etiquetas usamos la función `parTable()`

```
parTable(model1.e)
```

```
##    id     lhs op     rhs user block group free ustart exo label plabel
## 1   1  visual =~      x1    1     1     1    0      1   0         .p1.
## 2   2  visual =~      x2    1     1     1    1     NA   0  .p2.   .p2.
## 3   3  visual =~      x3    1     1     1    2     NA   0  .p3.   .p3.
## 4   4 textual =~      x4    1     1     1    0      1   0         .p4.
## 5   5 textual =~      x5    1     1     1    3     NA   0  .p5.   .p5.
## 6   6 textual =~      x6    1     1     1    4     NA   0  .p6.   .p6.
## 7   7   speed =~      x7    1     1     1    0      1   0         .p7.
## 8   8   speed =~      x8    1     1     1    5     NA   0  .p8.   .p8.
## 9   9   speed =~      x9    1     1     1    6     NA   0  .p9.   .p9.
## 10 10      x1 ~~      x1    0     1     1    7     NA   0        .p10.
## 11 11      x2 ~~      x2    0     1     1    8     NA   0        .p11.
## 12 12      x3 ~~      x3    0     1     1    9     NA   0        .p12.
## 13 13      x4 ~~      x4    0     1     1   10     NA   0        .p13.
## 14 14      x5 ~~      x5    0     1     1   11     NA   0        .p14.
## 15 15      x6 ~~      x6    0     1     1   12     NA   0        .p15.
## 16 16      x7 ~~      x7    0     1     1   13     NA   0        .p16.
## 17 17      x8 ~~      x8    0     1     1   14     NA   0        .p17.
## 18 18      x9 ~~      x9    0     1     1   15     NA   0        .p18.
## 19 19  visual ~~  visual    0     1     1   16     NA   0        .p19.
## 20 20 textual ~~ textual    0     1     1   17     NA   0        .p20.
## 21 21   speed ~~   speed    0     1     1   18     NA   0        .p21.
## 22 22  visual ~~ textual    0     1     1   19     NA   0        .p22.
## 23 23  visual ~~   speed    0     1     1   20     NA   0        .p23.
## 24 24 textual ~~   speed    0     1     1   21     NA   0        .p24.
## 25 25      x1 ~1            0     1     1   22     NA   0 .p25.  .p25.
## 26 26      x2 ~1            0     1     1   23     NA   0 .p26.  .p26.
## 27 27      x3 ~1            0     1     1   24     NA   0 .p27.  .p27.
## 28 28      x4 ~1            0     1     1   25     NA   0 .p28.  .p28.
## 29 29      x5 ~1            0     1     1   26     NA   0 .p29.  .p29.
## 30 30      x6 ~1            0     1     1   27     NA   0 .p30.  .p30.
## 31 31      x7 ~1            0     1     1   28     NA   0 .p31.  .p31.
## 32 32      x8 ~1            0     1     1   29     NA   0 .p32.  .p32.
## 33 33      x9 ~1            0     1     1   30     NA   0 .p33.  .p33.
## 34 34  visual ~1            0     1     1    0      0   0        .p34.
## 35 35 textual ~1            0     1     1    0      0   0        .p35.
## 36 36   speed ~1            0     1     1    0      0   0        .p36.
## 37 37  visual =~      x1    1     2     2    0      1   0        .p37.
## 38 38  visual =~      x2    1     2     2   31     NA   0  .p2.  .p38.
## 39 39  visual =~      x3    1     2     2   32     NA   0  .p3.  .p39.
## 40 40 textual =~      x4    1     2     2    0      1   0        .p40.
## 41 41 textual =~      x5    1     2     2   33     NA   0  .p5.  .p41.
## 42 42 textual =~      x6    1     2     2   34     NA   0  .p6.  .p42.
## 43 43   speed =~      x7    1     2     2    0      1   0        .p43.
## 44 44   speed =~      x8    1     2     2   35     NA   0  .p8.  .p44.
## 45 45   speed =~      x9    1     2     2   36     NA   0  .p9.  .p45.
## 46 46      x1 ~~      x1    0     2     2   37     NA   0        .p46.
## 47 47      x2 ~~      x2    0     2     2   38     NA   0        .p47.
## 48 48      x3 ~~      x3    0     2     2   39     NA   0        .p48.
## 49 49      x4 ~~      x4    0     2     2   40     NA   0        .p49.
```

```
## 50 50      x5 ~~      x5    0     2     2   41     NA   0        .p50.
## 51 51      x6 ~~      x6    0     2     2   42     NA   0        .p51.
## 52 52      x7 ~~      x7    0     2     2   43     NA   0        .p52.
## 53 53      x8 ~~      x8    0     2     2   44     NA   0        .p53.
## 54 54      x9 ~~      x9    0     2     2   45     NA   0        .p54.
## 55 55  visual ~~  visual    0     2     2   46     NA   0        .p55.
## 56 56 textual ~~ textual    0     2     2   47     NA   0        .p56.
## 57 57   speed ~~   speed    0     2     2   48     NA   0        .p57.
## 58 58  visual ~~ textual    0     2     2   49     NA   0        .p58.
## 59 59  visual ~~   speed    0     2     2   50     NA   0        .p59.
## 60 60 textual ~~   speed    0     2     2   51     NA   0        .p60.
## 61 61      x1 ~1           0     2     2   52     NA   0 .p25.  .p61.
## 62 62      x2 ~1           0     2     2   53     NA   0 .p26.  .p62.
## 63 63      x3 ~1           0     2     2   54     NA   0 .p27.  .p63.
## 64 64      x4 ~1           0     2     2   55     NA   0 .p28.  .p64.
## 65 65      x5 ~1           0     2     2   56     NA   0 .p29.  .p65.
## 66 66      x6 ~1           0     2     2   57     NA   0 .p30.  .p66.
## 67 67      x7 ~1           0     2     2   58     NA   0 .p31.  .p67.
## 68 68      x8 ~1           0     2     2   59     NA   0 .p32.  .p68.
## 69 69      x9 ~1           0     2     2   60     NA   0 .p33.  .p69.
## 70 70  visual ~1           0     2     2   61     NA   0        .p70.
## 71 71 textual ~1           0     2     2   62     NA   0        .p71.
## 72 72   speed ~1           0     2     2   63     NA   0        .p72.
## 73 73    .p2. ==   .p38.    2     0     0    0     NA   0
## 74 74    .p3. ==   .p39.    2     0     0    0     NA   0
## 75 75    .p5. ==   .p41.    2     0     0    0     NA   0
## 76 76    .p6. ==   .p42.    2     0     0    0     NA   0
## 77 77    .p8. ==   .p44.    2     0     0    0     NA   0
## 78 78    .p9. ==   .p45.    2     0     0    0     NA   0
## 79 79   .p25. ==   .p61.    2     0     0    0     NA   0
## 80 80   .p26. ==   .p62.    2     0     0    0     NA   0
## 81 81   .p27. ==   .p63.    2     0     0    0     NA   0
## 82 82   .p28. ==   .p64.    2     0     0    0     NA   0
## 83 83   .p29. ==   .p65.    2     0     0    0     NA   0
## 84 84   .p30. ==   .p66.    2     0     0    0     NA   0
## 85 85   .p31. ==   .p67.    2     0     0    0     NA   0
## 86 86   .p32. ==   .p68.    2     0     0    0     NA   0
## 87 87   .p33. ==   .p69.    2     0     0    0     NA   0
##    start    est    se
## 1  1.000  1.000 0.000
## 2  0.769  0.576 0.101
## 3  1.186  0.798 0.112
## 4  1.000  1.000 0.000
## 5  1.237  1.120 0.066
## 6  0.865  0.932 0.056
## 7  1.000  1.000 0.000
## 8  1.227  1.130 0.145
## 9  0.827  1.009 0.132
## 10 0.698  0.555 0.139
## 11 0.752  1.296 0.158
## 12 0.673  0.944 0.136
## 13 0.660  0.445 0.069
## 14 0.854  0.502 0.082
## 15 0.487  0.263 0.050
```

```
## 16 0.585  0.888 0.120
## 17 0.476  0.541 0.095
## 18 0.489  0.654 0.096
## 19 0.050  0.796 0.172
## 20 0.050  0.879 0.131
## 21 0.050  0.322 0.082
## 22 0.000  0.410 0.095
## 23 0.000  0.178 0.066
## 24 0.000  0.180 0.062
## 25 4.941  5.001 0.090
## 26 5.984  6.151 0.077
## 27 2.487  2.271 0.083
## 28 2.823  2.778 0.087
## 29 3.995  4.035 0.096
## 30 1.922  1.926 0.079
## 31 4.432  4.242 0.073
## 32 5.563  5.630 0.072
## 33 5.418  5.465 0.069
## 34 0.000  0.000 0.000
## 35 0.000  0.000 0.000
## 36 0.000  0.000 0.000
## 37 1.000  1.000 0.000
## 38 0.896  0.576 0.101
## 39 1.155  0.798 0.112
## 40 1.000  1.000 0.000
## 41 0.991  1.120 0.066
## 42 0.962  0.932 0.056
## 43 1.000  1.000 0.000
## 44 1.282  1.130 0.145
## 45 0.895  1.009 0.132
## 46 0.659  0.654 0.128
## 47 0.613  0.964 0.123
## 48 0.537  0.641 0.101
## 49 0.629  0.343 0.062
## 50 0.671  0.376 0.073
## 51 0.640  0.437 0.067
## 52 0.531  0.625 0.095
## 53 0.547  0.434 0.088
## 54 0.526  0.522 0.086
## 55 0.050  0.708 0.160
## 56 0.050  0.870 0.131
## 57 0.050  0.505 0.115
## 58 0.000  0.427 0.097
## 59 0.000  0.329 0.082
## 60 0.000  0.236 0.073
## 61 4.930  5.001 0.090
## 62 6.200  6.151 0.077
## 63 1.996  2.271 0.083
## 64 3.317  2.778 0.087
## 65 4.712  4.035 0.096
## 66 2.469  1.926 0.079
## 67 3.921  4.242 0.073
## 68 5.488  5.630 0.072
## 69 5.327  5.465 0.069
```

```
## 70 0.000 -0.148 0.122
## 71 0.000  0.576 0.117
## 72 0.000 -0.177 0.090
## 73 0.000  0.000 0.000
## 74 0.000  0.000 0.000
## 75 0.000  0.000 0.000
## 76 0.000  0.000 0.000
## 77 0.000  0.000 0.000
## 78 0.000  0.000 0.000
## 79 0.000  0.000 0.000
## 80 0.000  0.000 0.000
## 81 0.000  0.000 0.000
## 82 0.000  0.000 0.000
## 83 0.000  0.000 0.000
## 84 0.000  0.000 0.000
## 85 0.000  0.000 0.000
## 86 0.000  0.000 0.000
## 87 0.000  0.000 0.000
```

Esto nos indica que las constantes de x3 y x7 tienen la mayor contribución al desajuste. En otras palabras, estos parámetros no son invariantes entre los grupos. Pero... también significa que si los "liberamos" (i.e. no los dejamos o forzamos a que sean iguales) podremos tener invarianza globlal parcial.

NOTA: Piensen en lo que harían si no fuera tan claro que dos parámetros son el problema. Ustedes creen que dos personas usarían los mismos criterios de decisión?

Invarianza escalar parcial

Para dejar *libres* a las constantes de x3 y x7 podemos usar la función `group.partial`:

```
model1.ep<- cfa(ModeloHS, data=HolzingerSwineford1939, group="school",
                group.equal=c("loadings","intercepts"), group.partial=c("x3~1", "x7~1"))
```

Ahora checamos si nuestro modelo es igual de bueno que el configural:

```
anova(model1,model1.ep)
```

```
## Chi-Squared Difference Test
##
##           Df    AIC    BIC  Chisq Chisq diff Df diff Pr(>Chisq)
## model1    48 7484.4 7706.8 115.85
## model1.ep 58 7478.0 7663.3 129.42     13.571      10     0.1935
```