Psychology 454: Latent Variable Modeling

further adventures with lavaan

Department of Psychology
Northwestern University
Evanston, Illinois USA

February, 2011

Outline

Latent Variable Modeling programs

- Commercial programs
  - LISREL/PRELIS (Jöreskog, 1978; Joreskog & Sorbom, 1993; Jöreskog & Sörbom, 1999) http://www.ssicentral.com/lisrel/techdocs/IPUG.pdf Users Manual
  - EQS (Bentler, 1995)
  - AMOS Arbuckle (1989, 1994) http://spss.wikia.com/wiki/SEM_(structural_equation_modeling)_-_Amos Amos wiki
  - MPLUS (Muthén & Muthén, 2007) http://www.statmodel.com/ugexcerpts.shtml User's guide with examples
- Open source
  - Mx (Neale, 1994)
  - OpenMx
  - sem (Fox, 2009)
  - lavaan (Rosseel, 2010)

lavaan 0.4-7 from CRAN. Description from the user's guide

- The lavaan package is free open-source software. This means (among other things) that there is no warranty whatsoever.
- The numerical results of the lavaan package are typically very close, if not identical, to the results of the commercial package Mplus. If you wish to compare the results with other SEM packages, you can use the optional argument mimic="EQS" when calling the cfa, sem or growth functions
- The lavaan package is not finished yet. But it is already very useful for most users, or so we hope. There are a number of known minor issues and some features are simply not implemented yet.
- Some important features that are currently not available in lavaan are:
  - support for categorical/censored variables
  - support for discrete latent variables (mixture models)
  - support for hierarchical/multilevel datasets

Data sets available in lavaan

- HolzingerSwineford1939: A data frame with 301 observations of 15 variables.
    - The classic Holzinger and Swineford (1939) dataset consists of mental ability test scores of seventh- and eighth-grade children from two different schools (Pasteur and Grant-White). In the original dataset (available in the MBESS package), there are scores for 26 tests. However, a smaller subset with 9 variables is more widely used in the literature (for example in Joreskog's 1969 paper, which also uses the 145 subjects from the Grant-White school only
- PoliticalDemocracy: A data frame of 75 observations of 11 variables.
    - The ''famous'' Industrialization and Political Democracy dataset. This dataset is used throughout Bollen's 1989 book (see pages 12, 17, 36 in chapter 2, pages 228 and following in chapter 7, pages 321 and following in chapter 8). The dataset contains various measures of political democracy and industrialization in developing countries.

item syntax

```
Regression
y ~ f1 + f2 + x1 + x2
f1 ~ f2 + f3
f2 ~ f3 + x1 + x2

Latent variables
f1 =~ y1 + y2 + y3
f2 =~ y4 + y5 + y6
f3 =~ y7 + y8 + y9 + y10

Variances and covariances
y1 ~~ y1
y1 ~~ y2
f1 ~~ f2

Intercepts
y1 ~ 1
f1 ~ 1
```

Entering the model syntax as a string literal

```
myModel <- ' # regressions
y1 + y2 ~ f1 + f2 + x1 + x2
f1 ~ f2 + f3
f2 ~ f3 + x1 + x2
# latent variable definitions
f1 =~ y1 + y2 + y3
f2 =~ y4 + y5 + y6
f3 =~ y7 + y8 +
y9 + y10
# variances and covariances
y1 ~~ y1
y1 ~~ y2
f1 ~~ f2
# intercepts
y1 ~ 1
f1 ~ 1
'
```

The HolzingerSwineford data set

```
HS.model <- '
visual =~ x1 + x2 + x3
 textual =~ x4 + x5 + x6
 speed =~ x7 + x8 + x9
 '
fit <- cfa(HS.model, data = HolzingerSwineford1939)

summary(fit, fit.measures = TRUE)
lavaan.diagram(fit)  #need to use the newer function
```

Holzinger Swineford analysis

```
Lavaan (0.4-7) converged normally after 35 iterations

  Number of observations                           301

  Estimator                                         ML
  Minimum Function Chi-square                   85.306
  Degrees of freedom                                24
  P-value                                        0.000

Chi-square test baseline model:

  Minimum Function Chi-square                  918.852
  Degrees of freedom                                36
  P-value                                        0.000

Full model versus baseline model:

  Comparative Fit Index (CFI)                    0.931
  Tucker-Lewis Index (TLI)                       0.896

Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)              -3737.745
  Loglikelihood unrestricted model (H1)      -3695.092

  Number of free parameters                         21
  Akaike (AIC)                                7517.490
  Bayesian (BIC)                              7595.339
  Sample-size adjusted Bayesian (BIC)         7528.739
```

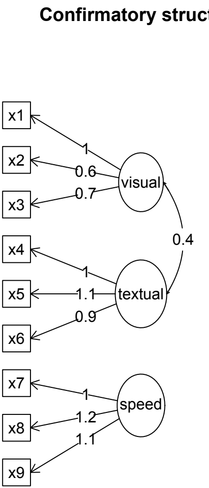
```
Root Mean Square Error of Approximation:

  RMSEA                                              0.092
  90 Percent Confidence Interval          0.071      0.114
  P-value RMSEA <= 0.05                              0.001

Standardized Root Mean Square Residual:

  SRMR                                               0.065

Parameter estimates:

  Information                                     Expected
  Standard Errors                                 Standard

                   Estimate  Std.err  Z-value  P(>|z|)
Latent variables:
  visual =~
    x1                1.000
    x2                0.554    0.100    5.554    0.000
    x3                0.729    0.109    6.685    0.000
  textual =~
    x4                1.000
    x5                1.113    0.065   17.014    0.000
    x6                0.926    0.055   16.703    0.000
  speed =~
    x7                1.000
    x8                1.180    0.165    7.152    0.000
    x9                1.082    0.151    7.155    0.000
```

more parameters

```
Covariances:
  visual ~~
    textual           0.408    0.074    5.552    0.000
    speed             0.262    0.056    4.660    0.000
  textual ~~
    speed             0.173    0.049    3.518    0.000
```

```
Variances:
    x1                0.549    0.114    4.833    0.000
    x2                1.134    0.102   11.146    0.000
    x3                0.844    0.091    9.317    0.000
    x4                0.371    0.048    7.778    0.000
    x5                0.446    0.058    7.642    0.000
    x6                0.356    0.043    8.277    0.000
    x7                0.799    0.081    9.823    0.000
    x8                0.488    0.074    6.573    0.000
    x9                0.566    0.071    8.003    0.000
    visual            0.809    0.145    5.564    0.000
    textual           0.979    0.112    8.737    0.000
    speed             0.384    0.086    4.451    0.000
```

Graphic output using (revised) lavaan.diagram



Redo with alternative parameterization

```
HS.model <- '
visual =~ x1 + x2 + x3
 textual =~ x4 + x5 + x6
 speed =~ x7 + x8 + x9
 '
fit <- cfa(HS.model, data = HolzingerSwineford1939,std.ov=TRUE,std.lv=TRUE)

summary(fit, fit.measures = TRUE)
lavaan.diagram(fit)  #need to use the newer function
```

```
Lavaan (0.4-7) converged normally after 21 iterations

  Number of observations                           301

  Estimator                                         ML
  Minimum Function Chi-square                   85.306
  Degrees of freedom                                24
  P-value                                        0.000

Chi-square test baseline model:

  Minimum Function Chi-square                  918.852
  Degrees of freedom                                36
  P-value                                        0.000

Full model versus baseline model:

  Comparative Fit Index (CFI)                    0.931
  Tucker-Lewis Index (TLI)                       0.896

Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)              -3422.624
  Loglikelihood unrestricted model (H1)      -3379.971

  Number of free parameters                         21
  Akaike (AIC)                                6887.248
  Bayesian (BIC)                              6965.097
  Sample-size adjusted Bayesian (BIC)         6898.497
```

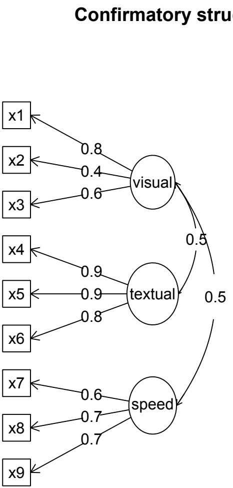
```
Root Mean Square Error of Approximation:

  RMSEA                                          0.092
  90 Percent Confidence Interval          0.071  0.114
  P-value RMSEA <= 0.05                          0.001

Standardized Root Mean Square Residual:

  SRMR                                           0.065

Parameter estimates:

  Information                                 Expected
  Standard Errors                             Standard

                   Estimate  Std.err  Z-value  P(>|z|)
Latent variables:
  visual =~
    x1                0.771    0.069   11.127    0.000
    x2                0.423    0.066    6.429    0.000
    x3                0.580    0.066    8.817    0.000
  textual =~
    x4                0.850    0.049   17.474    0.000
    x5                0.854    0.049   17.576    0.000
    x6                0.837    0.049   17.082    0.000
  speed =~
    x7                0.569    0.064    8.903    0.000
    x8                0.722    0.065   11.090    0.000
    x9                0.664    0.064   10.305    0.000
```

A simple confirmatory analysis

```
Covariances:
  visual ~~
    textual           0.459    0.064    7.189    0.000
    speed             0.471    0.073    6.461    0.000
  textual ~~
    speed             0.283    0.069    4.117    0.000
```

```
Variances:
    x1                0.403    0.083    4.833    0.000
    x2                0.818    0.073   11.146    0.000
    x3                0.660    0.071    9.317    0.000
    x4                0.274    0.035    7.779    0.000
    x5                0.268    0.035    7.642    0.000
    x6                0.297    0.036    8.277    0.000
    x7                0.673    0.069    9.823    0.000
    x8                0.476    0.072    6.573    0.000
    x9                0.556    0.069    8.003    0.000
    visual            1.000
    textual           1.000
    speed             1.000
```

The standardized solution to the Holzinger Swineford 1939 problem



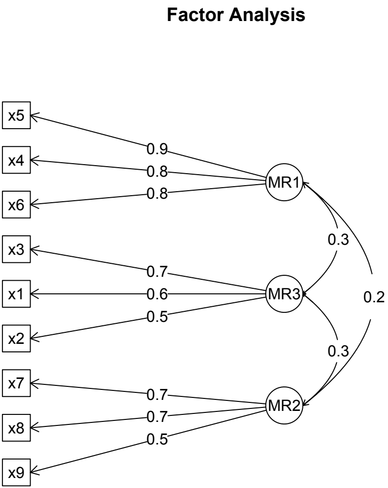
EFA of the Holzinger Swineford 1939 problem

```
f3 <- fa(HolzingerSwineford1939[7:15],3)
 f3
 diagram(f3,cut=.1)
```

```
Factor Analysis using method =  minres
Call: fa(r = HolzingerSwineford1939[7:15],
                nfactors = 3)
Standardized loadings based upon
         correlation matrix
     MR1   MR3   MR2   h2   u2
x1  0.19  0.60  0.03 0.49 0.51
x2  0.04  0.51 -0.12 0.25 0.75
x3 -0.07  0.69  0.02 0.46 0.54
x4  0.84  0.02  0.01 0.72 0.28
x5  0.89 -0.07  0.01 0.76 0.24
x6  0.81  0.08 -0.01 0.69 0.31
x7  0.04 -0.15  0.72 0.50 0.50
x8 -0.03  0.10  0.70 0.53 0.47
x9  0.03  0.37  0.46 0.46 0.54

                 MR1  MR3  MR2
SS loadings     2.24 1.34 1.28
Proportion Var 0.25 0.15 0.14
Cumulative Var 0.25 0.40 0.54

 With factor correlations of
     MR1  MR3  MR2
MR1 1.00 0.33 0.22
MR3 0.33 1.00 0.27
MR2 0.22 0.27 1.00
```

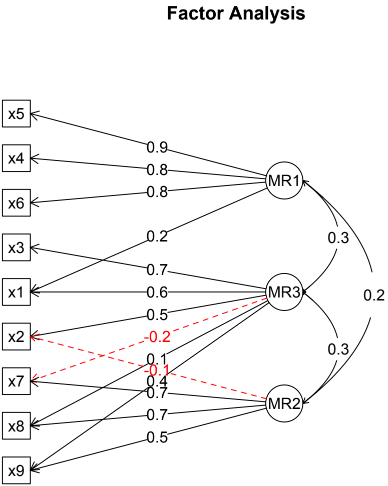
```
Test of the hypothesis that 3 factors are sufficient.

The degrees of freedom for the null model are  36  and the
objective function was  3.05 with Chi Square of  904.1
The degrees of freedom for the model are 12  and the
               objective function was  0.08

The root mean square of the residuals is  0.01
The df corrected root mean square of the residuals is  0.03
The number of observations was  301  with
                     Chi Square =  22.38  with prob <  0.034

Tucker Lewis Index of factoring reliability =  0.964
RMSEA index =  0.055  and the 90 % confidence intervals are
                     0.054 0.06
BIC =  -46.11
Fit based upon off diagonal values = 1
Measures of factor score adequacy
                                                MR1  MR3  MR2
Correlation of scores with factors             0.94 0.84 0.85
Multiple R square of scores with factors       0.89 0.71 0.72
Minimum correlation of possible factor scores  0.78 0.42 0.45
```

Holzinger Swineford EFA – compare with CFA



Holzinger Swineford EFA – simple is FALSE

```
diagram(f3,cut=.1,simple=FALSE)
```



Fixing parameters to simply models

- Perhaps the greatest power of SEM type programs is the ability to fix parameters or to force equality constraints.
    - EFA allows all parameters to vary
    - CFA allows only certain parameters to vary
- Can fix covariances to be zero
- Can fix different paths to be equal

Two ways of fixing the covariances to be 0

```
HS.ortho <- '
# three-factor model
visual =~ x1 + x2 + x3
textual =~ x4 + x5 + x6
speed =~ NA*x7 + x8 + x9
# orthogonal factors
visual ~~ 0*speed
textual ~~ 0*speed
# fix variance of speed factor
speed ~~ 1*speed'
fit.hs.ortho <- cfa(HS.ortho, data=HolzingerSwineford1939,std.ov=TRUE,std.lv=TRUE)

or (if they are all to be zero)

HS.model <- ' visual =~ x1 + x2 + x3
 textual =~ x4 + x5 + x6
 speed =~ x7 + x8 + x9 '
 fit.HS.ortho <- cfa(HS.model, data=HolzingerSwineford1939, orthogonal=TRUE)
```

```
Lavaan (0.4-7) converged normally after 20 iterations

  Number of observations                           301

  Estimator                                         ML
  Minimum Function Chi-square                  117.946
  Degrees of freedom                                26
  P-value                                        0.000
```

Not as good a fit

```
> summary(fit.hs.ortho,fit.measures=TRUE)

Lavaan (0.4-7) converged normally after 20 iterations

  Number of observations                           301

  Estimator                                         ML
  Minimum Function Chi-square                  117.946
  Degrees of freedom                                26
  P-value                                        0.000

Chi-square test baseline model:

  Minimum Function Chi-square                  918.852
  Degrees of freedom                                36
  P-value                                        0.000

Full model versus baseline model:

  Comparative Fit Index (CFI)                    0.896
  Tucker-Lewis Index (TLI)                       0.856

Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)              -3438.944
  Loglikelihood unrestricted model (H1)      -3379.971

  Number of free parameters                         19
  Akaike (AIC)                                6915.889
  Bayesian (BIC)                              6986.324
  Sample-size adjusted Bayesian (BIC)         6926.067
```

```
Root Mean Square Error of Approximation:

  RMSEA                                          0.108
  90 Percent Confidence Interval          0.089  0.129
  P-value RMSEA <= 0.05                          0.000

Standardized Root Mean Square Residual:

  SRMR                                           0.125

Parameter estimates:

  Information                                 Expected
  Standard Errors                             Standard

                   Estimate  Std.err  Z-value  P(>|z|)
Latent variables:
  visual =~
    x1                0.777    0.075   10.376    0.000
    x2                0.430    0.067    6.423    0.000
    x3                0.568    0.069    8.270    0.000
  textual =~
    x4                0.851    0.049   17.491    0.000
    x5                0.853    0.049   17.545    0.000
    x6                0.837    0.049   17.079    0.000
  speed =~
    x7                0.607    0.067    9.040    0.000
    x8                0.800    0.073   10.899    0.000
    x9                0.560    0.066    8.509    0.000
```

Fixing parameters - starting values and equality constraints

```
Covariances:
  visual ~~
    speed              0.000
  textual ~~
    speed              0.000
  visual ~~
    textual            0.461    0.064    7.195    0.000
```

```
Variances:
    speed              1.000
    x1                 0.394    0.095    4.155    0.000
    x2                 0.811    0.074   10.965    0.000
    x3                 0.675    0.074    9.085    0.000
    x4                 0.273    0.035    7.735    0.000
    x5                 0.269    0.035    7.662    0.000
    x6                 0.297    0.036    8.263    0.000
    x7                 0.629    0.073    8.650    0.000
    x8                 0.357    0.094    3.794    0.000
    x9                 0.683    0.071    9.640    0.000
    visual             1.000
    textual            1.000
```

Fixing starting values

(If you have a problem with a solution, you can help it if you give it a reasonable starting location.)

```
visual =~ x1 + start(0.8)*x2 + start(1.2)*x3
textual =~ x4 + start(0.5)*x5 + start(1.0)*x6
speed =~ x7 + start(0.7)*x8 + start(1.8)*x9
```

This technique works for all SEM programs (although details vary). The reason to give good starting values is that the search optimization can get bogged down in the wrong part of the parameter space.

Automatic naming

```
> model <- '
+ # latent variable definitions
+ ind60 =~ x1 + x2 + x3
+ dem60 =~ y1 + y2 + y3 + y4
+ dem65 =~ y5 + y6 + y7 + y8
+ # regressions
+ dem60 ~ ind60
+ dem65 ~ ind60 + dem60
+ # residual (co)variances
+ y1 ~~ y5
+ y2 ~~ y4 + y6
+ y3 ~~ y7
+ y4 ~~ y8
+ y6 ~~ y8
+ '
> fit <- sem(model, data=PoliticalDemocracy)
coef(fit)
```

```
ind60=~x2 ind60=~x3 dem60=~y2 dem60=~y3 dem60=~y4 dem65=~y6
2.180 1.819 1.257 1.058 1.265 1.186
dem65=~y7 dem65=~y8 dem60~ind60 dem65~ind60 dem65~dem60 y1~~y5
1.280 1.266 1.483 0.572 0.837 0.624
y2~~y4 y2~~y6 y3~~y7 y4~~y8 y6~~y8 x1~~x1
1.313 2.153 0.795 0.348 1.356 0.082
x2~~x2 x3~~x3 y1~~y1 y2~~y2 y3~~y3 y4~~y4
0.120 0.467 1.891 7.373 5.068 3.148
y5~~y5 y6~~y6 y7~~y7 y8~~y8 ind60~~ind60 dem60~~dem60
2.351 4.954 3.431 3.254 0.448 3.956
dem65~~dem65
0.172
```

Specifying the name

```
> model <- '
+ # latent variable definitions
+ ind60 =~ x1 + x2 + label("myLabel")*x3
+ dem60 =~ y1 + y2 + y3 + y4
+ dem65 =~ y5 + y6 + y7 + y8
+ # regressions
+ dem60 ~ ind60
+ dem65 ~ ind60 + dem60
+ # residual (co)variances
+ y1 ~~ y5
+ y2 ~~ y4 + y6
+ y3 ~~ y7
+ y4 ~~ y8
+ y6 ~~ y8
+ '
```

Using names to specify equality constraints

```
visual =~ x1 + x2 + equal("visual=~x2")*x3
textual =~ x4 + x5 + x6
speed =~ x7 + x8 + x9
```

Estimate the means

```
HS.model.means <- '
# three-factor model
visual =~ x1 + x2 + x3
textual =~ x4 + x5 + x6
speed =~ x7 + x8 + x9
# intercepts
x1 ~ 1
x2 ~ 1
x3 ~ 1
x4 ~ 1
x5 ~ 1
x6 ~ 1
x7 ~ 1
x8 ~ 1
x9 ~ 1'
fit.means <- cfa(HS.model.means,data = HolzingerSwineford1939,std.lv=
TRUE)

or
fit <- cfa(HS.model, data = HolzingerSwineford1939, meanstructure = TRUE)
summary(fit.means,fit.measures=TRUE)
```

Just show the parameter estimates

```
                   Estimate  Std.err  Z-value  P(>|z|)
Latent variables:
  visual =~
    x1                0.900    0.081   11.127    0.000
    x2                0.498    0.077    6.429    0.000
    x3                0.656    0.074    8.817    0.000
  textual =~
    x4                0.990    0.057   17.474    0.000
    x5                1.102    0.063   17.576    0.000
    x6                0.917    0.054   17.082    0.000
  speed =~
    x7                0.619    0.070    8.903    0.000
    x8                0.731    0.066   11.090    0.000
    x9                0.670    0.065   10.305    0.000

Covariances:
  visual ~~
    textual           0.459    0.064    7.189    0.000
    speed             0.471    0.073    6.461    0.000
  textual ~~
    speed             0.283    0.069    4.117    0.000

Intercepts:
    x1                4.936    0.067   73.473    0.000
    x2                6.088    0.068   89.855    0.000
    x3                2.250    0.065   34.579    0.000
    x4                3.061    0.067   45.694    0.000
    x5                4.341    0.074   58.452    0.000
    x6                2.186    0.063   34.667    0.000
    x7                4.186    0.063   66.766    0.000
    x8                5.527    0.058   94.854    0.000
    x9                5.374    0.058   92.546    0.000
    visual            0.000
    textual           0.000
    speed             0.000

Variances:
    x1                0.549    0.114    4.833    0.000
    x2                1.134    0.102   11.146    0.000
    x3                0.844    0.091    9.317    0.000
    x4                0.371    0.048    7.779    0.000
    x5                0.446    0.058    7.642    0.000
    x6                0.356    0.043    8.277    0.000
    x7                0.799    0.081    9.823    0.000
    x8                0.488    0.074    6.573    0.000
    x9                0.566    0.071    8.003    0.000
    visual            1.000
    textual           1.000
    speed             1.000
```

More useful if we want to fix some intercepts to be different from others

```
# three-factor model
visual =~ x1 + x2 + x3
textual =~ x4 + x5 + x6
speed =~ x7 + x8 + x9
# intercepts with fixed values
x1 ~ 0.5*1
x2 ~ 0.5*1
x3 ~ 0.5*1
x4 ~ 0.5*1
```

Analyzing multiple groups

- When studying differences in ages, gender, school, it is useful to be able to model them separately, but to get an overall goodness of fit.
  - Does a basic structure hold in different groups?
- More importantly, we can ask if the parameters in the two groups are the same. That is, we can add equality constraints.
- We can examine equality of the loadings, equality of the covariances, equality of the mean structure.

```
HS.model <- ' visual =~ x1 + x2 + x3
textual =~ x4 + x5 + x6
speed =~ x7 + x8 + x9 '
fit <- cfa(HS.model, data=HolzingerSwineford1939, group="school")
summary(fit)
```

```
Lavaan (0.4-7) converged normally after 60 iterations

  Number of observations per group
  Pasteur                                          156
  Grant-White                                      145

  Estimator                                         ML
  Minimum Function Chi-square                  115.851
  Degrees of freedom                                48
  P-value                                        0.000

Chi-square for each group:

  Pasteur                                       64.309
  Grant-White                                   51.542

Parameter estimates:

  Information                                 Expected
  Standard Errors                             Standard
```

Multiple groups

```
Group 1 [Pasteur]:

                   Estimate  Std.err  Z-value  P(>|z|)
Latent variables:
  visual =~
    x1                1.000
    x2                0.394    0.122    3.220    0.001
    x3                0.570    0.140    4.076    0.000
  textual =~
    x4                1.000
    x5                1.183    0.102   11.613    0.000
    x6                0.875    0.077   11.421    0.000
  speed =~
    x7                1.000
    x8                1.125    0.277    4.057    0.000
    x9                0.922    0.225    4.104    0.000

Covariances:
  visual ~~
    textual           0.479    0.106    4.531    0.000
    speed             0.185    0.077    2.397    0.017
  textual ~~
    speed             0.182    0.069    2.628    0.009

Variances:
    x1                0.298    0.232    1.286    0.198
    x2                1.334    0.158    8.464    0.000
    x3                0.989    0.136    7.271    0.000
    x4                0.425    0.069    6.138    0.000
    x5                0.456    0.086    5.292    0.000
    x6                0.290    0.050    5.780    0.000
    x7                0.820    0.125    6.580    0.000
    x8                0.510    0.116    4.406    0.000
    x9                0.680    0.104    6.516    0.000
    visual            1.097    0.276    3.967    0.000
```

```
Group 2 [Grant-White]:

                   Estimate  Std.err  Z-value  P(>|z|)
Latent variables:
  visual =~
    x1                1.000
    x2                0.736    0.155    4.760    0.000
    x3                0.925    0.166    5.584    0.000
  textual =~
    x4                1.000
    x5                0.990    0.087   11.418    0.000
    x6                0.963    0.085   11.377    0.000
  speed =~
    x7                1.000
    x8                1.226    0.187    6.569    0.000
    x9                1.058    0.165    6.429    0.000

Covariances:
  visual ~~
    textual           0.408    0.098    4.153    0.000
    speed             0.276    0.076    3.639    0.000
  textual ~~
    speed             0.222    0.073    3.022    0.003

Variances:
    x1                0.715    0.126    5.675    0.000
    x2                0.899    0.123    7.339    0.000
    x3                0.557    0.103    5.409    0.000
    x4                0.315    0.065    4.870    0.000
    x5                0.419    0.072    5.812    0.000
    x6                0.406    0.069    5.880    0.000
    x7                0.600    0.091    6.584    0.000
    x8                0.401    0.094    4.248    0.000
    x9                0.535    0.089    6.010    0.000
```

Multiple groups, multiple constraints

- Can constrain a single parameter to be equal across groups
  - Use the naming convention and the equal comand
- Can constrain equivalent parameters across groups to be equal (group.equal)

Equal parameters across groups

```
HS.model <- ' visual =~ x1 + x2 + x3
 textual =~ x4 + x5 + x6
 speed =~ x7 + x8 + x9 '
fit <- cfa(HS.model, data=HolzingerSwineford1939, group="school",
 group.equal=c("loadings"),std.ov=TRUE,std.lv=TRUE)
 summary(fit)
```

```
Lavaan (0.4-7) converged normally after 27 iterations

  Number of observations per group
  Pasteur                                            156
  Grant-White                                        145

  Estimator                                           ML
  Minimum Function Chi-square                    122.862
  Degrees of freedom                                  57
  P-value                                          0.000

Chi-square for each group:

  Pasteur                                         68.598
  Grant-White                                     54.264
```

Multiple groups

```
Group 1 [Pasteur]:

                   Estimate  Std.err  Z-value  P(>|z|)
Latent variables:
  visual =~
    x1                0.737    0.066   11.100    0.000
    x2                0.448    0.065    6.888    0.000
    x3                0.625    0.065    9.658    0.000
  textual =~
    x4                0.841    0.049   17.127    0.000
    x5                0.843    0.049   17.207    0.000
    x6                0.830    0.049   16.826    0.000
  speed =~
    x7                0.597    0.063    9.490    0.000
    x8                0.736    0.064   11.559    0.000
    x9                0.644    0.063   10.221    0.000

Covariances:
  visual ~~
    textual           0.467    0.087    5.369    0.000
    speed             0.343    0.109    3.149    0.002
  textual ~~
    speed             0.333    0.094    3.527    0.000

Variances:
    x1                0.421    0.095    4.438    0.000
    x2                0.823    0.102    8.040    0.000
    x3                0.634    0.096    6.635    0.000
    x4                0.316    0.052    6.137    0.000
    x5                0.267    0.048    5.590    0.000
    x6                0.299    0.050    6.042    0.000
    x7                0.698    0.098    7.095    0.000
    x8                0.528    0.100    5.303    0.000

Group 2 [Grant-White]:

                   Estimate  Std.err  Z-value  P(>|z|)
Latent variables:
  visual =~
    x1                0.737    0.066   11.100    0.000
    x2                0.448    0.065    6.888    0.000
    x3                0.625    0.065    9.658    0.000
  textual =~
    x4                0.841    0.049   17.127    0.000
    x5                0.843    0.049   17.207    0.000
    x6                0.830    0.049   16.826    0.000
  speed =~
    x7                0.597    0.063    9.490    0.000
    x8                0.736    0.064   11.559    0.000
    x9                0.644    0.063   10.221    0.000

Covariances:
  visual ~~
    textual           0.537    0.085    6.315    0.000
    speed             0.530    0.097    5.477    0.000
  textual ~~
    speed             0.331    0.093    3.557    0.000

Variances:
    x1                0.479    0.095    5.043    0.000
    x2                0.759    0.098    7.758    0.000
    x3                0.567    0.089    6.399    0.000
    x4                0.260    0.048    5.399    0.000
    x5                0.302    0.052    5.817    0.000
    x6                0.312    0.052    5.998    0.000
    x7                0.577    0.084    6.894    0.000
    x8                0.379    0.081    4.703    0.000
```

Do the measures measure the same construct across groups?

- Is the configuration the same?
  - Most abstract level of invariance – "are the arrows the same"
- Weak invariance – are the loadings the same?
- Strong invariance – equal loadings + intercepts

Testing for measurement invariance

```
measurementInvariance(HS.model, data = HolzingerSwineford1939,
 group = "school")
Measurement invariance tests:

Model 1: configural invariance:
   chisq       df    pvalue      cfi     rmsea       bic
 115.851   48.000    0.000    0.923    0.097 7604.094

Model 2: weak invariance (equal loadings):
   chisq       df    pvalue      cfi     rmsea       bic
 124.044   54.000    0.000    0.921    0.093 7578.043

[Model 1 versus model 2]
  delta.chisq      delta.df delta.p.value      delta.cfi
        8.192         6.000         0.224          0.002

Model 3: strong invariance (equal loadings + intercepts):
   chisq       df    pvalue      cfi     rmsea       bic
 164.103   60.000    0.000    0.882    0.107 7686.588

[Model 1 versus model 3]
  delta.chisq      delta.df delta.p.value      delta.cfi
       48.251        12.000         0.000          0.041

[Model 2 versus model 3]
  delta.chisq      delta.df delta.p.value      delta.cfi
       40.059         6.000         0.000          0.038

Model 4: equal loadings + intercepts + means:
   chisq       df    pvalue      cfi     rmsea       bic
 204.605   63.000    0.000    0.854    0.122 7709.969
```

Demo.growth : A toy data set

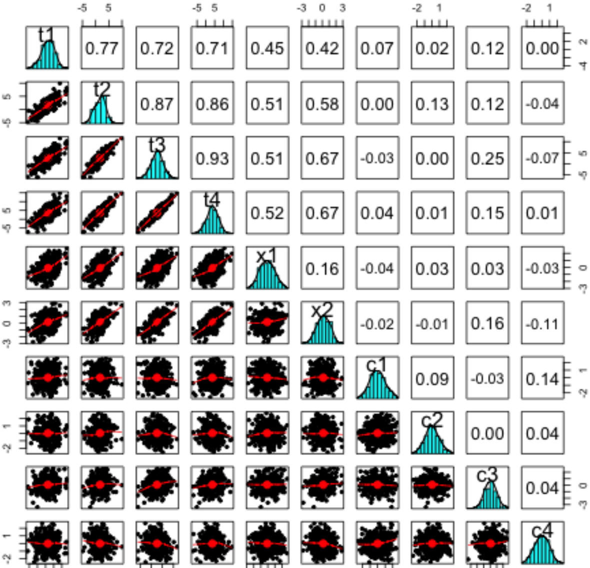
```
data(Demo.growth)
 describe(Demo.growth)
```

	var	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se
t1	1	400	0.59	1.58	0.67	0.64	1.50	-4.35	5.21	9.56	-0.18	0.23	0.08
t2	2	400	1.67	2.13	1.88	1.69	2.10	-4.86	9.95	14.81	-0.02	0.48	0.11
t3	3	400	2.59	2.72	2.73	2.62	2.62	-6.02	11.53	17.55	-0.14	0.46	0.14
t4	4	400	3.64	3.38	3.72	3.69	3.14	-7.34	14.72	22.06	-0.13	0.44	0.17
x1	5	400	-0.09	1.03	-0.08	-0.11	1.11	-2.82	2.72	5.54	0.08	-0.33	0.05
x2	6	400	0.14	0.96	0.13	0.15	1.01	-2.83	2.88	5.71	-0.12	0.01	0.05
c1	7	400	0.01	0.99	-0.03	-0.01	0.98	-2.58	2.57	5.14	0.11	-0.28	0.05
c2	8	400	0.03	0.95	0.01	0.02	0.87	-2.54	2.71	5.25	0.13	-0.07	0.05
c3	9	400	0.07	0.93	0.07	0.06	0.93	-3.40	2.61	6.01	-0.02	0.38	0.05
c4	10	400	-0.02	0.92	-0.01	-0.02	0.95	-2.45	2.65	5.11	0.02	-0.21	0.05

Data.growth: A toy data set

1. t1 Measured value at time point 1
2. t2 Measured value at time point 2
3. t3 Measured value at time point 3
4. t4 Measured value at time point 4
5. x1 Predictor 1 influencing intercept and slope
6. x2 Predictor 2 influencing intercept and slope
7. c1 Time-varying covariate time point 1
8. c2 Time-varying covariate time point 2
9. c3 Time-varying covariate time point 3
10. c4 Time-varying covariate time point 4

Growth: SPLOM



Fitting a growth model to the toy problem

```
model <- ' i =~ 1*t1 + 1*t2 + 1*t3 + 1*t4
  s =~ 0*t1 + 1*t2 + 2*t3 + 3*t4 '
 fit <- growth(model, data=Demo.growth)
summary(fit)
```

```
Lavaan (0.4-7) converged normally after 43 iterations

  Number of observations                           400

  Estimator                                         ML
  Minimum Function Chi-square                    8.069
  Degrees of freedom                                 5
  P-value                                        0.152

Parameter estimates:

  Information                                 Expected
  Standard Errors                             Standard

                 Estimate  Std.err  Z-value  P(>|z|)
```

Growth model with parameter values

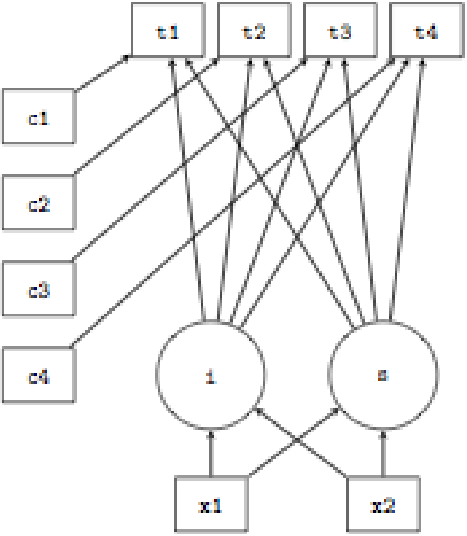
```
Latent variables:
  i =~
    t1               1.000
    t2               1.000
    t3               1.000
    t4               1.000
  s =~
    t1               0.000
    t2               1.000
    t3               2.000
    t4               3.000

Covariances:
  i ~~
    s                0.618    0.071    8.686    0.000
```

```
Intercepts:
    t1               0.000
    t2               0.000
    t3               0.000
    t4               0.000
    i                0.615    0.077    8.007    0.000
    s                1.006    0.042   24.076    0.000

Variances:
    t1               0.595    0.086    6.944    0.000
    t2               0.676    0.061   11.061    0.000
    t3               0.635    0.072    8.761    0.000
    t4               0.508    0.124    4.090    0.000
    i                1.932    0.173   11.194    0.000
    s                0.587    0.052   11.336    0.000
```

From the lavaan manual

- "Technically, the growth function is almost identical to the sem function. But a meanstructure is automatically assumed, and the observed intercepts are fixed to zero by default, while the latent variable intercepts and means are freely estimated.
- A slightly more complex model adds two regressors (x1 and x2) that influence the latent growth factors.
- In addition, a time-varying covariate that influences the outcome measure at the four time points has been added to the model.
- A graphical representation of this model together with the corresponding lavaan syntax is presented".

A growth model



Linear growth with time-varying covariates

```
# a linear growth model with a time-varying covariate
model <- '
# intercept and slope with fixed coefficients
i =~ 1*t1 + 1*t2 + 1*t3 + 1*t4
s =~ 0*t1 + 1*t2 + 2*t3 + 3*t4
# regressions
i ~ x1 + x2
s ~ x1 + x2
# time-varying covariates
t1 ~ c1
t2 ~ c2
t3 ~ c3
t4 ~ c4
'
fit <- growth(model, data=Demo.growth)
summary(fit)
```

```
Lavaan (0.4-7) converged normally after 40 iterations

  Number of observations                          400

  Estimator                                        ML
  Minimum Function Chi-square                  26.059
  Degrees of freedom                               21
```

Parameters of the growth model

```
                   Estimate  Std.err  Z-value  P(>|z|)
Latent variables:
  i =~
    t1                1.000
    t2                1.000
    t3                1.000
    t4                1.000
  s =~
    t1                0.000
    t2                1.000
    t3                2.000
    t4                3.000

Regressions:
  i ~
    x1                0.608    0.060   10.134    0.000
    x2                0.604    0.064    9.412    0.000
  s ~
    x1                0.262    0.029    9.198    0.000
    x2                0.522    0.031   17.083    0.000
  t1 ~
    c1                0.143    0.050    2.883    0.004
  t2 ~
    c2                0.289    0.046    6.295    0.000
  t3 ~
    c3                0.328    0.044    7.361    0.000
  t4 ~
    c4                0.330    0.058    5.655    0.000

Covariances:
  i ~~
    s                 0.075    0.040    1.855    0.064

Intercepts:
    t1                0.000
    t2                0.000
    t3                0.000
    t4                0.000
    i                 0.580    0.062    9.368    0.000
    s                 0.958    0.029   32.552    0.000

Variances:
    t1                0.580    0.080    7.230    0.000
    t2                0.596    0.054   10.969    0.000
    t3                0.481    0.055    8.745    0.000
    t4                0.535    0.098    5.466    0.000
    i                 1.079    0.112    9.609    0.000
    s                 0.224    0.027    8.429    0.000
```

Modifying a model

- There are many reasons a model does not fit.
  - In particular, some paths may be badly fit (usually because they were ignored).
  - How much will a parameter change (Expected Parameter Change) if a parameter is adjusted.

Modification indices for the HS problem

```
fit <- cfa(HS.model, data = HolzingerSwineford1939)
 mi <- modindices(fit)
mi[mi$op == "=~", ]
```

```
       lhs op rhs     mi    epc sepc.lv sepc.all
1   visual =~  x1     NA     NA      NA       NA
2   visual =~  x2  0.000  0.000   0.000    0.000
3   visual =~  x3  0.000  0.000   0.000    0.000
4   visual =~  x4  1.211  0.077   0.069    0.059
5   visual =~  x5  7.441 -0.210  -0.189   -0.147
6   visual =~  x6  2.843  0.111   0.100    0.092
7   visual =~  x7 18.631 -0.422  -0.380   -0.349
8   visual =~  x8  4.295 -0.210  -0.189   -0.187
9   visual =~  x9 36.411  0.577   0.519    0.515
10 textual =~  x1  8.903  0.350   0.347    0.297
11 textual =~  x2  0.017 -0.011  -0.011   -0.010
12 textual =~  x3  9.151 -0.272  -0.269   -0.238
13 textual =~  x4     NA     NA      NA       NA
14 textual =~  x5  0.000  0.000   0.000    0.000
15 textual =~  x6  0.000  0.000   0.000    0.000
16 textual =~  x7  0.098 -0.021  -0.021   -0.019
17 textual =~  x8  3.359 -0.121  -0.120   -0.118
18 textual =~  x9  4.796  0.138   0.137    0.136
19   speed =~  x1  0.014  0.024   0.015    0.013
20   speed =~  x2  1.580 -0.198  -0.123   -0.105
21   speed =~  x3  0.716  0.136   0.084    0.075
22   speed =~  x4  0.003 -0.005  -0.003   -0.003
23   speed =~  x5  0.201 -0.044  -0.027   -0.021
24   speed =~  x6  0.273  0.044   0.027    0.025
25   speed =~  x7     NA     NA      NA       NA
26   speed =~  x8  0.000  0.000   0.000    0.000
27   speed =~  x9  0.000  0.000   0.000    0.000
```

The fitted model

```
fit <- cfa(HS.model, data = HolzingerSwineford1939)
fitted(fit)
```

```
$cov
   x1    x2    x3    x4    x5    x6    x7    x8    x9
x1 1.358
x2 0.448 1.382
x3 0.590 0.327 1.275
x4 0.408 0.226 0.298 1.351
x5 0.454 0.252 0.331 1.090 1.660
x6 0.378 0.209 0.276 0.907 1.010 1.196
x7 0.262 0.145 0.191 0.173 0.193 0.161 1.183
x8 0.309 0.171 0.226 0.205 0.228 0.190 0.453 1.022
x9 0.284 0.157 0.207 0.188 0.209 0.174 0.415 0.490 1.015

$mean
x1 x2 x3 x4 x5 x6 x7 x8 x9
 0  0  0  0  0  0  0  0  0
```

Examine the raw residuals

```
fit <- cfa(HS.model, data = HolzingerSwineford1939)
 resid(fit)

$cov
   x1     x2     x3     x4     x5     x6     x7     x8     x9
x1  0.000
x2 -0.041  0.000
x3 -0.010  0.124  0.000
x4  0.097 -0.017 -0.090  0.000
x5 -0.014 -0.040 -0.219  0.008  0.000
x6  0.077  0.038 -0.032 -0.012  0.005  0.000
x7 -0.177 -0.242 -0.103  0.046 -0.050 -0.017  0.000
x8 -0.046 -0.062 -0.013 -0.079 -0.047 -0.024  0.082  0.000
x9  0.175  0.087  0.167  0.056  0.086  0.062 -0.042 -0.032  0.000

$mean
x1 x2 x3 x4 x5 x6 x7 x8 x9
 0  0  0  0  0  0  0  0  0
```

Examine the standardized residuals

```
fit <- cfa(HS.model, data = HolzingerSwineford1939)
 resid(fit, type = "standardized")
```

```
$cov
   x1     x2     x3     x4     x5     x6     x7     x8     x9
x1  0.000
x2 -2.196  0.000
x3 -1.199  2.692  0.000
x4  2.465 -0.283 -1.948     NA
x5 -0.362 -0.610 -4.443  0.856     NA
x6  2.032  0.661 -0.701     NA  0.633  0.000
x7 -3.787 -3.800 -1.882  0.839 -0.837 -0.321     NA
x8 -1.456 -1.137 -0.305 -2.049 -1.100 -0.635  3.804  0.000
x9  4.062  1.517  3.328  1.237  1.723  1.436 -2.771     NA  0.000
```

Many more examples

- The MPlus manual has data sets that may be explored with lavaan code.
    - Chapter 3: Regression and Path Analysis
    - Chapter 5: Confirmatory factor analysis and structural equation modeling
    - Chapter 6: Growth modeling
- LISREL manual also has suitable examples

SEM-AMOS wiki a warning

- It may seem odd to begin with a warning, but the popular misuse and misinterpretation of Structural Equation Modeling is so widespread that users of this wiki should be aware of some of the issues involved before they begin. While this warning is overly brief, you can follow-up these issues and more in the Further Reading section of this article.
- A number of these issues also apply to Confirmatory Factor Analysis. While Structural Equation Modeling has been popular in recent years to test the degree of fit between a proposed structural model and the emergent structure of the data, the perceived superiority of the technique is waning.
- Aside from the fact that the results of Structural Equation Modeling are often poorly reported, the conclusions drawn do not typically grasp the limitations of the technique.

SEM-AMOS wiki a warning page 2

- The most obvious, and some ways the most critical issue is that of incorrectly inferring a particular configuration of causal relationships from correlational data. This mistake can be illustrated with the simplest of all structural examples – that of 2 variables (variable A and B). If we ignore the additional complexity of latent structure, the number of possible causal structures is 4. Clearly, the number of possible models grows exponentially as the number of variables grows. In this example, the 4 possible causal models in this example are:
  - A causes B;
  - B causes A;
  - A and B cause each other (a recursive model);
  - finally, A and B are unrelated.

SEM-AMOS wiki warning page 3

- If A and B are indeed significantly correlated, it is likely that the first 3 models will be supported by significant fit statistics. If this is the case, what has been proven?
- Which of the 3 supported models is the correct model? What makes matters worse is that we have not even conclusively ruled out the last model. It is still possible that the correlation between A and B was spurious.
- To reinforce a maxim that most people know, but fail to apply to Structural Equation Modeling – you can not determine causation from correlation.
- Yet in most cases, researchers only test one or two models out of all the myriad of potential models, poorly report their results, then proclaim confirmation of their model (implying the exclusion of all other possible models).

SEM-AMOS wiki warning page 4

- So what is the value of Structural Equation Modeling?
- If large correlational datasets are already available, and a large range of plausible models are assessed, the results can be valuable in conceiving an experimental study that can test the proposed causal relationships.

Arbuckle, J. L. (1989). AMOS: Analysis of moment structures. *The American Statistician*, *43*, 66.

Arbuckle, J. L. (1994). AMOS: Analysis of moment structures. *Psychometrika*, *59*, 135–137.

Bentler, P. M. (1995). *EQS structural equations program manual*. Encino, CA.: Multivariate Software, Inc.

Fox, J. (2009). *sem: Structural Equation Models*. R package version 0.9-15.

Jöreskog, K. (1978). Structural analysis of covariance and correlation matrices. *Psychometrika*, *43*(4), 443–477.

Joreskog, K. G. & Sorbom, D. (1993). *LISREL 8: Structural equation modeling with the SIMPLIS command language.* Lisrel 8: Lawrence Erlbaum Associates, Inc.

Jöreskog, K. G. & Sörbom, D. (1999). *LISREL 8: Structural equation modeling with the SIMPLIS command language.* Lincolnwood: Scientific Software International.

Muthén, L. & Muthén, B. (2007). *Mplus User's Guide* (Fifth Edition ed.). Los Angeles, CA: Muthén & Muthén.

Neale, M. C. (1994). *Mx: Statistical Modeling*. Box 710 MCV, Richmond, VA.: Department of Psychiatry.

Rosseel, Y. (2010). *lavaan: Latent Variable Analysis*. R package version 0.4-3.