lavaan: an R package for structural equation modeling and more

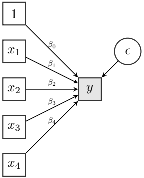
Yves Rosseel
Department of Data Analysis
Ghent University

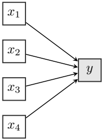
Psychoco 2011 – Tübingen

Overview

1. (gentle) introduction to structural equation modeling (SEM)
2. introducing the `lavaan` package
3. three small examples (cfa, sem, growth)
4. how does lavaan work?
5. future plans

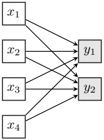
Univariate linear regression





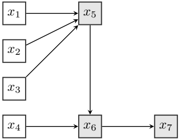
$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \beta_4 x_{i4} + \epsilon_i \quad (i = 1, 2, \ldots, n)$$

Multivariate regression



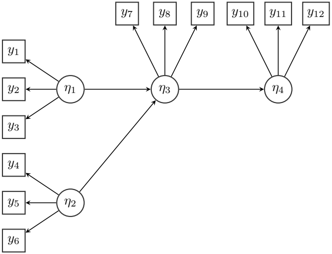
Path Analysis

- testing models of ‘causal’ relationships among observed variables
- all variables are observed (manifest)
- system of regression equations



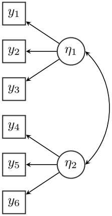
Structural Equation Modeling

- path analysis with latent variables



Measurement part only: confirmatory factor analysis (CFA)

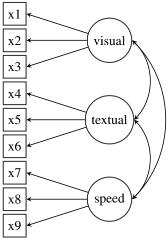
- factor analysis: representing the relationship between one or more latent variables and their (observed) indicators



Classic example CFA

- well-known dataset; based on Holzinger & Swineford (1939) data
- also analyzed by Jöreskog (1969)
- 9 observed ‘indicators’ measuring three ‘latent’ factors:
  - a ‘visual’ factor measured by x1, x2 and x3
  - a ‘textual’ factor measured by x4, x5 and x6
  - a ‘speed’ factor measured by x7, x8 and x9
- N=301
- we assume the three factors are correlated

Diagram of the model



Observed covariance matrix: $\mathbf{S}$

- n is the number of observed variables: $n = 9$
- observed covariance matrix:

	x1	x2	x3	x4	x5	x6	x7	x8	x9
x1	1.36								
x2	0.41	1.38							
x3	0.58	0.45	1.28						
x4	0.51	0.21	0.21	1.35					
x5	0.44	0.21	0.11	1.10	1.66				
x6	0.46	0.25	0.24	0.90	1.01	1.20			
x7	0.09	-0.10	0.09	0.22	0.14	0.14	1.18		
x8	0.26	0.11	0.21	0.13	0.18	0.17	0.54	1.02	
x9	0.46	0.24	0.37	0.24	0.30	0.24	0.37	0.46	1.02

- we want to 'explain' the observed correlations/covariances by postulating a number of latent variables (factors) and a corresponding factor structure
- we will 'rewrite' the $n(n+1)/2 = 45$ elements in the covariance matrix as a function a smaller number of 'free parameters' in the CFA model, summarized in a number of (typically sparse) matrices

The standard CFA model: matrix representation

- the classic LISREL representation uses three matrices (for CFA)
- the LAMBDA matrix contains the 'factor structure':

$$\mathbf{\Lambda} = \begin{bmatrix} x & 0 & 0 \\ x & 0 & 0 \\ x & 0 & 0 \\ 0 & x & 0 \\ 0 & x & 0 \\ 0 & x & 0 \\ 0 & 0 & x \\ 0 & 0 & x \\ 0 & 0 & x \end{bmatrix}$$

- the variances/covariances of the latent variables are summarized in the PSI matrix:

$$\mathbf{\Psi} = \begin{bmatrix} x & & \\ x & x & \\ x & x & x \end{bmatrix}$$

- what we can *not* explain by the set of common factors (the 'residual part' of the model) is written in the (typically diagonal) matrix THETA:

$$\mathbf{\Theta} = \begin{bmatrix} x & & & & & & & & \\ & x & & & & & & & \\ & & x & & & & & & \\ & & & x & & & & & \\ & & & & x & & & & \\ & & & & & x & & & \\ & & & & & & x & & \\ & & & & & & & x & \\ & & & & & & & & x \end{bmatrix}$$

- note that we have only 24 parameters (of which 21 are estimable)

The standard CFA model: parameter estimation

- in the standard CFA model, the 'implied' covariance matrix is:

$$\mathbf{\Sigma} = \mathbf{\Lambda}\mathbf{\Psi}\mathbf{\Lambda}' + \mathbf{\Theta}$$

- estimation problem: choose the 'free' parameters, so that the estimated implied covariance matrix ($\hat{\mathbf{\Sigma}}$) is 'as close as possible' to the observed covariance matrix $\mathbf{S}$
  - generalized (weighted) least-squares estimation
  - maximum likelihood estimation
- identification: we need to fix the 'scale' of the latent variables
  - for each factor: fix the loading of one indicator to 1.0
  - OR: fix the variance of the factors to 1.0 (=standardize the latent variables)

Software for SEM (commercial)

The big four

- LISREL
- EQS
- AMOS
- MPLUS

Others

- CALIS/TCALIS (SAS/Stat)
- SEPATH (Statistica)
- RAMONA (Systat)
- . . .

Software for SEM (non-commercial)

- Mx
- gllamm (Stata)
- . . .
- various R packages (sem, OpenMx, lavaan)

A short history of LISREL

- 1969: seminal paper by Karl Jöreskog: *A General Approach to Confirmatory Maximum Likelihood Factor Analysis*, published in Psychometrika
- 1970: Karl Jöreskog wrote the first FORTRAN program for CFA: ACOVS, later extended to ACOVSM, COFAMM, and eventually LISREL I (1972)
- 1972: LISREL I (LInear Structural RELationships) + LISREL II
- 1976: LISREL III (first commercial version?)
- 1978: LISREL IV
- 1981: LISREL V
- 1984: LISREL VI (as part of SPSS/X)
- 1989: LISREL 7 (as part of SPSS/PC)
- 1993: LISREL 8
- today: LISREL 8.8

What is lavaan?

- **lavaan** is an R package for latent variable analysis:
  - confirmatory factor analysis: function `cfa()`
  - structural equation modeling: function `sem()`
  - latent curve analysis / growth modeling: function `growth()`
  - (item response theory (IRT) models)
  - (latent class + mixture models)
  - (multilevel models)
- the **lavaan** package is developed to provide useRs, researchers and teachers a free, open-source, but commercial-quality package for latent variable modeling
- the long-term goal of **lavaan** is to implement all the state-of-the-art capabilities that are currently available in commercial packages

Current status of lavaan

- 1st public (CRAN) release of lavaan (0.3-1): May 2010
- 2nd public (CRAN) release of lavaan (0.4.7): Feb 2011
- webpage: `http://lavaan.org`
  - documentation: 'Introduction to lavaan' (about 25 pages)
  - overview of new features/changes, known issues and bugs/glitches
  - development versions

Why do we need lavaan?

- perhaps the best state-of-the-art software packages in this field are still closed-source and/or commerical:
  - commercial: LISREL, EQS, AMOS, MPLUS
  - free, but closed-source: Mx
  - free, but relying on third-party commercial software: `gllamm` (stata), OpenMx (the NPSOL solver)
- it seems unfortunate that new developments in this field are hindered by the lack of open source software that researchers can use to implement their newest ideas
- in addition, teaching these techniques to students was often complicated by the forced choice for one of these commercial packages

Related R packages

- **sem**
  - developer: John Fox (since 2001)
  - for a long time the only option in R
- **OpenMx**
  - 'advanced' structural equation modeling
  - developed at the University of Virginia (PI: Steven Boker)
  - Mx reborn
  - free, but the solver is (currently) not open-source
  - `http://openmx.psyc.virginia.edu/`
- interfaces between R and commercial packages:
  - `REQS`
  - `MplusAutomation`

Features of lavaan

1. lavaan is reliable and robust

- extensive testing before a 'public' release on CRAN
- no convergence problems (for admissible models)
- numerical results are very close (if not identical) to commercial packages:
  - Mplus (if `mimic="Mplus"`, default)
  - EQS (if `mimic="EQS"`)

2. lavaan is easy and intuitive to use

- the 'lavaan model syntax' allows users to express their models in a compact, elegant and useR-friendly way
- many 'default' options keep the model syntax clean and compact
- but the useR has full control (cfr. function `lavaan()`)

3. lavaan provides many advanced options

- full support for meanstructures and multiple groups
- several estimators are available (GLS, WLS, ML and variants)
- standard errors: using either observed or expected information
- support for nonnormal data: using robust standard errors and a scaled test statistic (Satorra-Bentler)
- support for missing data: direct ML (aka full information ML), with robust standard errors and a scaled test statistic (Yuan-Bentler)
- all gradients are computed analytically
- equality constraints (both within and across groups)
- ...

4. lavaan provides a wealth of information

- the `summary` gives a compact overview of the results
- the `fitMeasures` function provides a number of popular fit measures (CFI, TLI, RMSEA, SRMR, ...)
- the `modindices` function provides modification indices and corresponding expected parameter changes (EPCs)
- the `residuals` function provides raw, normalized and standardized residuals
- all computed information can be extracted from the fitted object using the `inspect` function
- `coef`, `fitted.values`, `vcov`, `predict`, `update`, `AIC`, `BIC`, ...

The 'lavaan model syntax'

- at the heart of the **lavaan** package is the 'model syntax': a formula-based description of the model to be estimated
- a distinction is made between four different formula types: 1) regression formulas, 2) latent variable definitions, 3) (co)variances, and 4) intercepts

1. regression formulas

- in the R environment, a regression formula has the following form:

```
y ~ x1 + x2 + x3 + x4
```

- in **lavaan**, a typical model is simply a set (or system) of regression formulas, where some variables (starting with an 'f' below) may be latent.
- for example:

```
y1 + y2 ~ f1 + f2 + x1 + x2
     f1 ~ f2 + f3
     f2 ~ f3 + x1 + x2
```

2. latent variable definitions

- if we have latent variables in any of the regression formulas, we need to 'define' them by listing their manifest indicators
- we do this by using the special operator "=~", which can be read as *is manifested by*
- for example:

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

3. (residual) variances and covariances

- variances and covariances are specified using a 'double tilde' operator
- for example:

```
y1 ~~ y1
y1 ~~ y2
f1 ~~ f2
```


4. intercepts

- intercepts are simply regression formulas with only an intercept (explicitly denoted by the number '1') as the only predictor
- for both observed and latent variables
- for example:

```
y1 ~ 1
f1 ~ 1
```

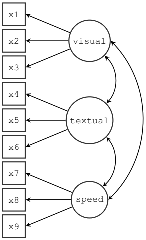

a complete description of a model: literal string

- enclose the model syntax by single quotes

```
> myModel <- ' # regressions
                y ~ 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
             '
```

- or put the syntax in a separate (text) file, and read it in using `readLines()`

Example 1: confirmatory factor analysis



lavaan model syntax

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

Fitting a model using the lavaan package

- from a useR point of view, fitting a model using **lavaan** consists of three steps:
  1. specify the model (using the model syntax)
  2. fit the model (using one of the functions `cfa`, `sem`, `growth`)
  3. see the results (using the `summary`, or other extractor functions)
- for example:

```
> # 1. specify the model
> HS.model <- ' visual  =~ x1 + x2 + x3
+               textual =~ x4 + x5 + x6
+               speed   =~ x7 + x8 + x9 '

> # 2. fit the model
> fit <- cfa(HS.model, data=HolzingerSwineford1939)

> # 3. display summary output
> summary(fit, fit.measures=TRUE, standardized=TRUE)
```


Output `summary(fit, fit.measures=TRUE, standardized=TRUE)`

```
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
```

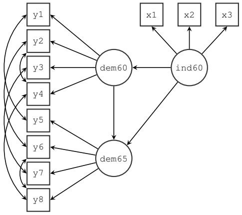


```
  Akaike (AIC)                                7517.490
  Bayesian (BIC)                              7595.339
  Sample-size adjusted Bayesian (BIC)         7528.739

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|)   Std.lv  Std.all
Latent variables:
  visual =~
    x1                1.000                               0.900    0.772
    x2                0.554    0.100    5.554    0.000    0.498    0.424
    x3                0.729    0.109    6.685    0.000    0.656    0.581
  textual =~
    x4                1.000                               0.990    0.852
    x5                1.113    0.065   17.014    0.000    1.102    0.855
```



```
    x6                0.926    0.055   16.703    0.000    0.917    0.838
  speed =~
    x7                1.000                               0.619    0.570
    x8                1.180    0.165    7.152    0.000    0.731    0.723
    x9                1.082    0.151    7.155    0.000    0.670    0.665

Covariances:
  visual ~~
    textual           0.408    0.074    5.552    0.000    0.459    0.459
    speed             0.262    0.056    4.660    0.000    0.471    0.471
  textual ~~
    speed             0.173    0.049    3.518    0.000    0.283    0.283

Variances:
    x1                0.549    0.114    4.833    0.000    0.549    0.404
    x2                1.134    0.102   11.146    0.000    1.134    0.821
    x3                0.844    0.091    9.317    0.000    0.844    0.662
    x4                0.371    0.048    7.778    0.000    0.371    0.275
    x5                0.446    0.058    7.642    0.000    0.446    0.269
    x6                0.356    0.043    8.277    0.000    0.356    0.298
    x7                0.799    0.081    9.823    0.000    0.799    0.676
    x8                0.488    0.074    6.573    0.000    0.488    0.477
    x9                0.566    0.071    8.003    0.000    0.566    0.558
    visual            0.809    0.145    5.564    0.000    1.000    1.000
    textual           0.979    0.112    8.737    0.000    1.000    1.000
    speed             0.384    0.086    4.451    0.000    1.000    1.000
```

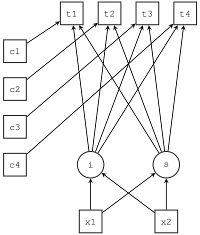
Example 2: structural equation model



lavaan model syntax

```
# latent variable definitions
  ind60 =~ x1 + x2 + x3
  dem60 =~ y1 + y2 + y3 + y4
  dem65 =~ y5 + y6 + y7 + y8

# regressions
  dem60 ~ ind60
  dem65 ~ ind60 + dem60

# residual covariances
  y1 ~~ y5
  y2 ~~ y4 + y6
  y3 ~~ y7
  y4 ~~ y8
  y6 ~~ y8
```


Example 3: growth curve model



lavaan model syntax

```
# 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 ~ label("a")*x1 + x2
  s ~ equal("a")*x1 +
      equal("i~x2")*x2

# time-varying covariates
  t1 ~ c1
  t2 ~ c2
  t3 ~ c3
  t4 ~ c4
```


How does lavaan work?

Step 1: From model syntax to 'list' representation

- model syntax is parsed by the function `lavaanify` which constructs a generic 'list' representation of the model
- decide which parameters are free or fixed, handle equality constraints and other user-requested modifications
- *optionally* add elements to make the model 'complete' (residual variances, covariances, intercepts, . . .)
- *optionally* fix the metric of latent variables
- everything is automatic if the functions `cfa`, `sem` or `growth` are used; nothing is done automatic if the function `lavaan` is used

```
> HS.model <- ' visual  =~ x1 + label("a")*x2 + x3
                textual =~ x4 + equal("a")*x5 + x6
                speed   =~ x7 + equal("a")*x8 + 0.75*x9 '

> User <- lavaanify(HS.model, auto.fix.first = TRUE, auto.var = TRUE,
                    auto.cov.lv.x = TRUE, orthogonal = TRUE)
```



```
> User
   id     lhs op     rhs user group free ustart fixed.x eq.id free.uncon
1   1  visual =~      x1    1     1    0   1.00       0     0          0
2   2  visual =~      x2    1     1    1     NA       0     2          1
3   3  visual =~      x3    1     1    2     NA       0     0          2
4   4 textual =~      x4    1     1    0   1.00       0     0          0
5   5 textual =~      x5    1     1    1     NA       0     2          3
6   6 textual =~      x6    1     1    3     NA       0     0          4
7   7   speed =~      x7    1     1    0   1.00       0     0          0
8   8   speed =~      x8    1     1    1     NA       0     2          5
9   9   speed =~      x9    1     1    0   0.75       0     0          0
10 10      x1 ~~      x1    0     1    4     NA       0     0          6
11 11      x2 ~~      x2    0     1    5     NA       0     0          7
12 12      x3 ~~      x3    0     1    6     NA       0     0          8
13 13      x4 ~~      x4    0     1    7     NA       0     0          9
14 14      x5 ~~      x5    0     1    8     NA       0     0         10
15 15      x6 ~~      x6    0     1    9     NA       0     0         11
16 16      x7 ~~      x7    0     1   10     NA       0     0         12
17 17      x8 ~~      x8    0     1   11     NA       0     0         13
18 18      x9 ~~      x9    0     1   12     NA       0     0         14
19 19  visual ~~  visual    0     1   13     NA       0     0         15
20 20 textual ~~ textual    0     1   14     NA       0     0         16
21 21   speed ~~   speed    0     1   15     NA       0     0         17
22 22  visual ~~ textual    0     1    0   0.00       0     0          0
23 23  visual ~~   speed    0     1    0   0.00       0     0          0
24 24 textual ~~   speed    0     1    0   0.00       0     0          0
```

Step 2: From 'list' representation to 'matrix' representation

- the 'list' representation is converted to a 'matrix' representation
- currently only the (all-y) LISREL representation is available
  - if no meanstructure, 4 matrices: LAMBDA, BETA, PSI, THETA
  - if meanstructure, two additional matrices: ALPHA, NU
- additional representations (EQS, RAM, . . .) can easily be added

```
> fit <- cfa(HS.model, data=HolzingerSwineford1939, orthogonal=TRUE)
> inspect(fit)

$lambda
   visual textul speed
x1      0      0     0
x2      1      0     0
x3      2      0     0
x4      0      0     0
x5      0      1     0
x6      0      3     0
x7      0      0     0
x8      0      0     1
x9      0      0     0
```



```
$theta
   x1 x2 x3 x4 x5 x6 x7 x8 x9
x1  4
x2  0  5
x3  0  0  6
x4  0  0  0  7
x5  0  0  0  0  8
x6  0  0  0  0  0  9
x7  0  0  0  0  0  0 10
x8  0  0  0  0  0  0  0 11
x9  0  0  0  0  0  0  0  0 12

$psi
        visual textul speed
visual  13
textual  0     14
speed    0      0     15
```


Step 3: fitting the model

- free parameters are estimated using unconstrained optimization
  - built-in optimizer: `nlminb`
  - analytical gradients
  - efficient conversion between:
    - 'matrix' representation (to compute the objective function and gradient)
    - 'vector' representation (to be used by the optimizer)
- optionally, (robust) standard errors are computed
- optionally, a (scaled) test statistic is computed
- a fitted object is created (S4 class 'lavaan')

Future plans

Support for categorical observed responses

- binary, ordinal, and limited-dependent (censored) observed responses
- using the 'limited-information' approach (eg polychoric correlations)
  - cf. Mplus WLSMV estimator
  - Gherard Arminger donated the source code of MECOSA to the lavaan project (written for GAUSS)
- using the maximum likelihood approach
  - entering the IRT world
  - lavaan as a front-end for IRT packages?

Support for discrete latent variables

- latent class and mixture models
- how should we implement this syntax-wise?

```
class(k=2)*c1 =~ y1 + y2 + y3 + y4
class(k=4)*c2 =~ y5 + y6 + y7
```

Support for hierarchical/multilevel data

Bayesian estimation

Export/import utilities

...

Thank you for your attention

`http://lavaan.org`