lavaan: an R package for structural equation modeling and more

Yves Rosseel
Department of Data Analysis
Ghent University

The R User Conference 2010

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

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
- numerical results are very close (if not identical) to commercial packages:
  - Mplus (if `mimic.Mplus=TRUE`, default)
  - EQS (if `mimic.Mplus=FALSE`)

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

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' (aka sandwish-type, Satorra-Bentler) standard errors and a scaled test statistic
- 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
- if requested, lavaan prints out a number of popular fit measures
- if requested, lavaan prints out modification indices and corresponding expected parameter changes (EPCs)
- all computed information can be extracted from the fitted object using the `inspect` function
- several extractor functions (`coef`, `fitted.values`, `residuals`, `vcov`) have been implemented

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:

```
 y ~ 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
```

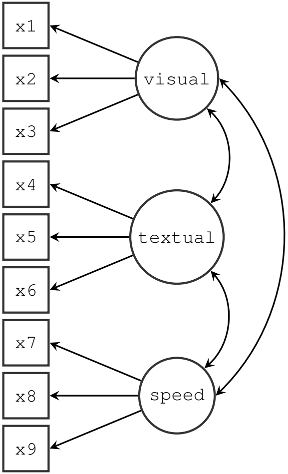
a complete description of a model: literal string

- 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)
```

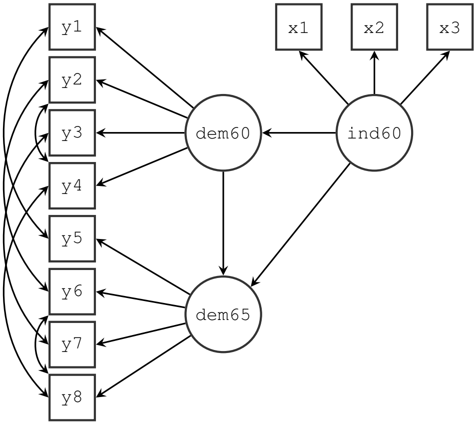
Output `summary(fit, fit.measures=TRUE, standardized=TRUE)`

```
Model converged normally after 35 iterations using ML

  Minimum Function Chi-square                     85.306
  Degrees of freedom                                  24
  P-value                                         0.0000

Chi-square test baseline model:

  Minimum Function Chi-square                    918.852
  Degrees of freedom                                  36
  P-value                                         0.0000

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

  Akaike (AIC)                                  7517.490
  Bayesian (BIC)                                7595.339
```

```
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


Model estimates:

                   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
```

```
Latent 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

Latent variances:
    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

Residual 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
```

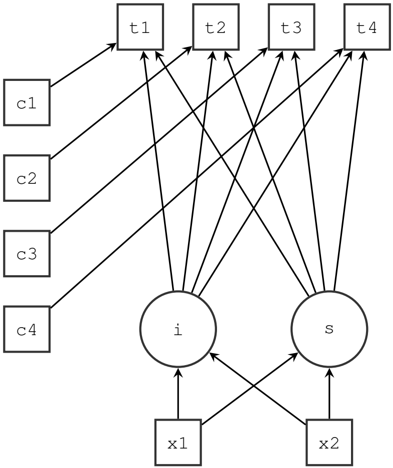
Example 2: structural equation model



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
```

Future plans

- support for categorical (binary and ordinal) and censored observed responses using the WLS(MV) approach (similar to Mplus, but based on the MECOSA source code)

- support for categorical (binary and ordinal) observed responses using the maximum likelihood approach (including IRT models)

- support for discrete latent variables (latent class and mixture models)

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

- support for hierarchical/multilevel data

- high-quality graphical output (eg. path diagrams) suitable for publishing

- various export/import/parsing routines to communicate with other packages

- ...

Thank you for your attention

http://lavaan.org