Open-source modern modeling software: the R package lavaan

Yves Rosseel
Department of Data Analysis
Ghent University – Belgium

Modern Modeling Methods Conference – 21 May 2013
University of Connecticut

contents

- part I:
  - software for modern modeling methods
  - software for SEM
  - lavaan is for statisticians, teachers and applied users
  - lavaan features (and missing features)
  - lavaan model syntax
- part II:
  - lavaan functions and options
  - lavaan and the (computational) history of SEM
  - future plans
  - discussion/questions

software for modern modeling methods: two approaches

- closed-source software
  - often highly advanced features
  - standalone (not integrated in a larger statistical package)
  - black box:
    no access to internal data, no access to internal functions
  - commercial (cost, licensing issues, no information about bugs)
- open-source software
  - in some areas, less advanced features
  - often integrated (R package, Stata module, Matlab toolbox)
  - open: full access to internal data, internal functions
  - flexibility: can be extended, modified or embedded in a larger pipeline
  - (academic) community

software for SEM: commercial – closed-source

- the big four:
  - LISREL
  - EQS
  - AMOS
  - Mplus
- SAS/Stat: proc CALIS, proc TCALIS
- SEPATH (Statistica), RAMONA (Systat), Stata 12
- Mx (free, closed-source)

software for SEM: non-commercial – open-source

- outside the R ecosystem: gllamm (Stata module), ...
- R packages:
  - sem
  - OpenMx
  - lavaan
  - lava
- interfaces between R and commercial packages:
  - REQS
  - MplusAutomation

what is lavaan?

- **lavaan** is an R package for latent variable analysis

- the long-term goal of **lavaan** is to implement all the state-of-the-art capabilities that are currently available in commercial packages

installing lavaan, finding documentation

- **lavaan** depends on the R project for statistical computing:

  http://www.r-project.org

- to install **lavaan**, simply start up an R session and type:

```
> install.packages("lavaan")
```

- more information about **lavaan**:

  http://lavaan.org

- the **lavaan** paper:

  Rosseel (2012). lavaan: an R package for structural equation modeling. *Journal of Statistical Software*, 48(2), 1–36.

- **lavaan** discussion group (mailing list)

  https://groups.google.com/d/forum/lavaan

why do we need lavaan?

1. **lavaan** is for statisticians working in the field of SEM
    - 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
2. **lavaan** is for teachers
    - teaching these techniques to students was often complicated by the forced choice for one of the commercial packages
3. **lavaan** is for applied researchers
    - keep it simple, provide all the features they need

lavaan is for statisticians working in the field of SEM (1)

- case-study: van de Schoot, R., Hoijtink, H. and Deković, M. (2010). Testing Inequality Constrained Hypotheses in SEM Models. *Structural Equation Modeling*, 17, 443-463

Testing Inequality Constrained Hypotheses in SEM Models

Rens van de Schoot and Herbert Hoijtink
Department of Methods and Statistics
Utrecht University, The Netherlands

Maja Deković
Department of Child and Adolescent Studies
Utrecht University, The Netherlands

Researchers often have expectations that can be expressed in the form of inequality constraints among the parameters of a structural equation model. It is currently not possible to test these so-called informative hypotheses in structural equation modeling software. We offer a solution to this problem using M*plus*. The hypotheses are evaluated using plug-in p values with a calibrated alpha level. The method is introduced and its utility is illustrated by means of an example.

van de Schoot (1)

- the problem: testing *informative* hypotheses in a SEM model

$$H_0 : \beta_{\text{male}} = \beta_{\text{female}} \quad \text{versus} \quad H_c : \beta_{\text{male}} > \beta_{\text{female}} \quad \text{(Type A)}$$

- van de Schoot et.al. (2010) present a frequentist solution: parametric bootstrap of the LRT to obtain a plug-in p-value
- seven steps:

1. estimate model under H_0 ($\beta_{\text{male}} = \beta_{\text{female}}$), save results
2. estimate model under H_c ($\beta_{\text{male}} > \beta_{\text{female}}$), save results
3. (manually) compute LRT.obs $(\text{logL}_c - \text{logL}_0)$
4. generate (R=1000) datasets under H_0
5. for each generated dataset, estimate H_0 and H_c model
6. (manually) compute (R=1000) LRT values,
7. compute plug-in p-value (proportion of LRTs $\geq$ LRT.obs)

van de Schoot (2)

- but under the null, the plug-in p-values should be (asymptotically) uniformly distributed; unfortunately, in this setting, this is not the case
- van de Schoot et.al. (2010) use a 'double bootstrap' procedure to calibrate the alpha (α) level:
  - each 'bootstrap sample' is again 'bootstrapped' (R2=1000)
  - we end up with R plug-in p-values, and the 5th percentile provides $\alpha^\star$ (the calibrated alpha)
  - we should use $\alpha^\star$ instead of the regular $\alpha = 0.05$
- challenge: try to do this using black box commercial SEM software ...
  - van de Schoot et.al. (2010) used Mplus, combined with a handful of custom R scripts (MplusAutomation was not available yet)

van de Schoot (3)

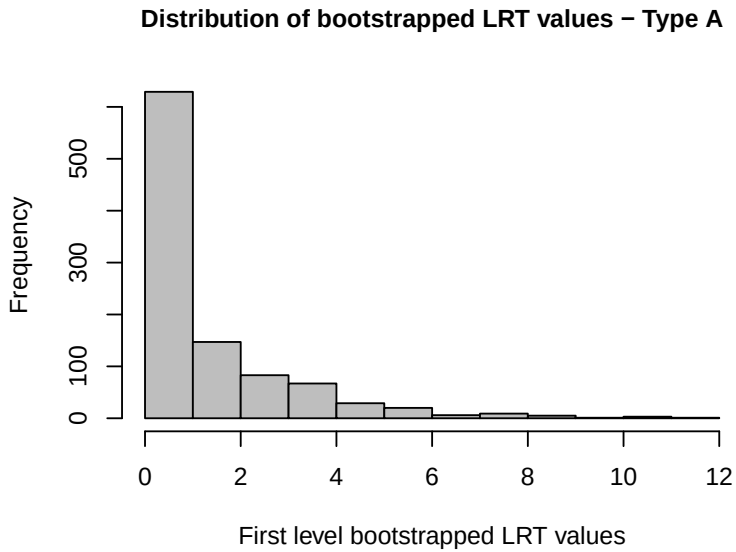
- what if we could do this in R?
- Leonard Vanbrabant wrote a high-level function to do all this in one simple step (available in **lavaan**)
- code snippet:

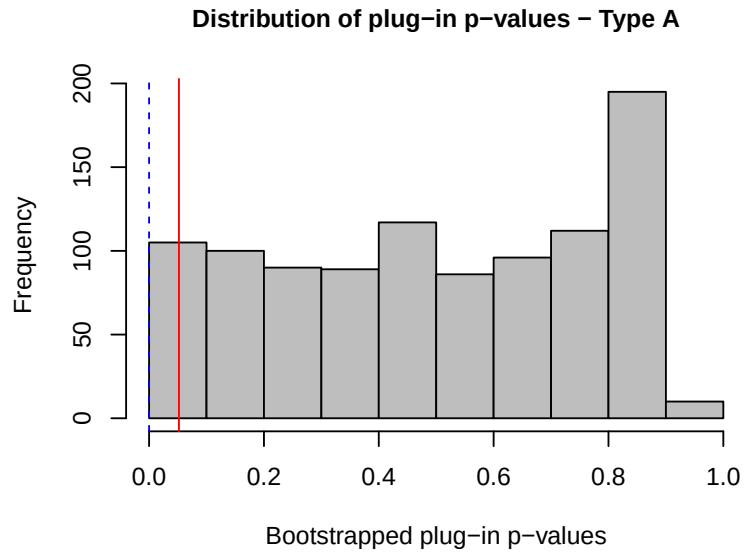
```
# simple regression model
model <- ' anti ~ b1*pos + b2*neg + b3*dis '

# inequality constraints (Hc)
constraints <- ' b1 < b3
                 b2 < b3 '

# compute plug-in p-value based on the double-bootstrap procedure
out <- InformativeTesting(model, data = social,
                          verbose = TRUE,
                          constraints = constraints,
                          R = 1000, type = "parametric",
                          double.bootstrap = "standard")
```

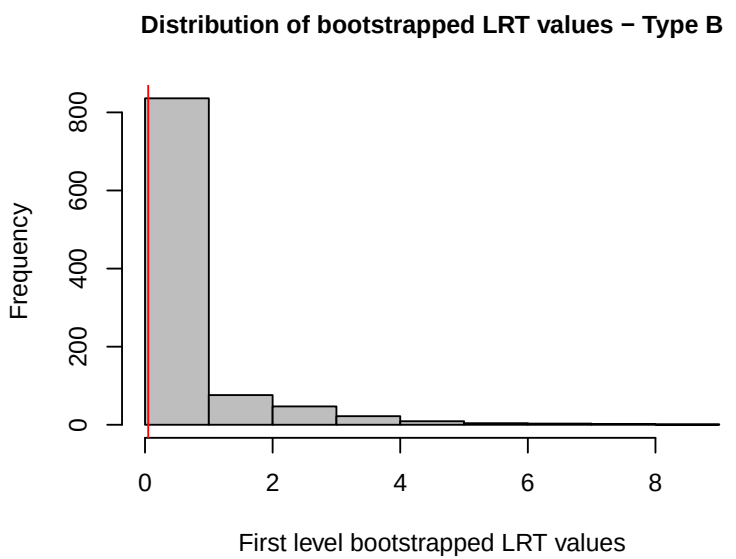
informativeTesting output

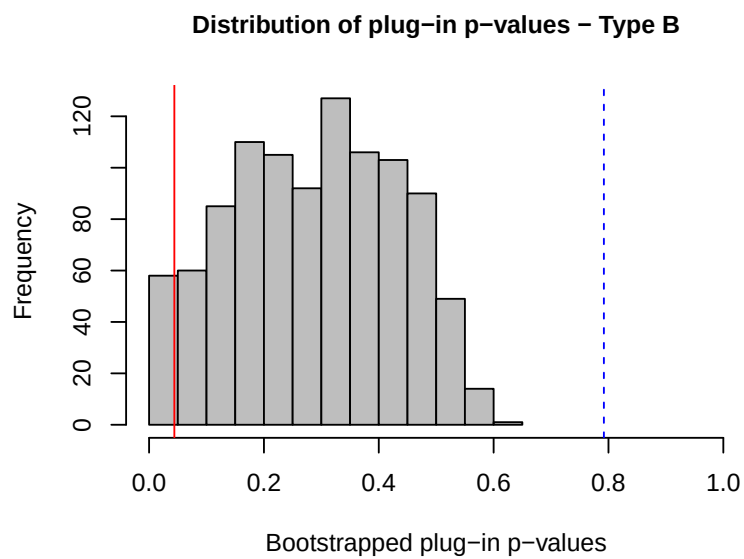
```
   Variable names in model        : anti pos neg dis
   Number of variables            : 4
   Number of groups               : 1
   Used sample size per group     : 596
   Used sample size               : 596
   Total sample size              : 603

   Estimator                      : ML
   Missing data                   : listwise
   Bootstrap method               : parametric
   Double bootstrap method        : standard


 Order Constrained Hypothesis Testing:

                              Type A            Type B
   ---------------------------------------------------
   LR statistic                47.98             00.05

   P-value                      0.00              0.43
   Adjusted alpha               0.05              0.04

   Adjusted p-value             0.00              0.79
   Alpha                        0.05              0.05

   Significance          Significant  Non-significant
```

informativeTesting plot



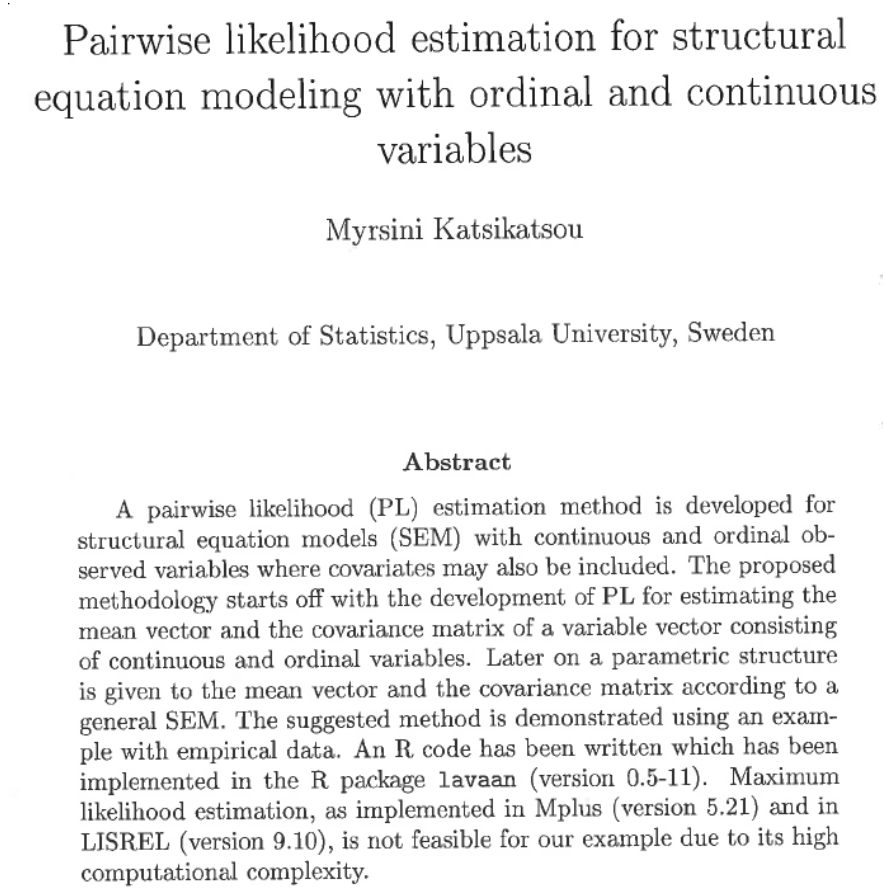






lavaan is for statisticians working in the field of SEM (2)

- case-study: Myrsini Katsikatsou (2012), Ph.D. dissertation, Paper 4



Pairwise likelihood estimation for structural equation modeling with ordinal and continuous variables

Myrsini Katsikatsou

Department of Statistics, Uppsala University, Sweden

Abstract

A pairwise likelihood (PL) estimation method is developed for structural equation models (SEM) with continuous and ordinal observed variables where covariates may also be included. The proposed methodology starts off with the development of PL for estimating the mean vector and the covariance matrix of a variable vector consisting of continuous and ordinal variables. Later on a parametric structure is given to the mean vector and the covariance matrix according to a general SEM. The suggested method is demonstrated using an example with empirical data. An R code has been written which has been implemented in the R package `lavaan` (version 0.5-11). Maximum likelihood estimation, as implemented in Mplus (version 5.21) and in LISREL (version 9.10), is not feasible for our example due to its high computational complexity.

- a lot of the infrastructure for the PML estimator is already present in commercial software (but inaccessible)

lavaan is for statisticians working in the field of SEM (3)

- case-study: Merkle, E.C. and Zeileis, A. (2013). Tests of measurment invariance without subgroups: a generalization of classical methods. *Psychometrika, 78*, 59–82

8. Computational Details

All results were obtained using the R system for statistical computing (R Development Core Team, 2012, version 2.15.0, employing the add-on packages lavaan 0.4-14 (Rosseel, 2012) and OpenMx 1.1.1-1785 (Boker, Neale, Maes, Wilde, Spiegel, Brick, & et al., 2011) for fitting of the factor analysis models and strucchange 1.4-7 (Zeileis, Leisch, Hornik, & Kleiber, 2002; Zeileis, 2006) for evaluating the parameter instability tests. R and the packages lavaan and strucchange are freely available under the General Public License 2 from the Comprehensive R Archive Network at http://CRAN.R-project.org/ while OpenMx is available under the Apache License 2.0 from http://OpenMx.psyc.virginia.edu/. R code for replication of our results is available at http://semtools.R-Forge.R-project.org/.

- again, computations were needed (eg. case-wise scores) that are commonly computed by commercial packages (but we can not extract the results)

lavaan is for teachers

- case-study:

  doctoral school SEM course in a Faculty of Psychology and Education Sciences; 40 Ph.D. students from 7 different departments; 5 departments had previously bought several licenses for either LISREL(2), EQS(1), AMOS(2)

  - which software package should we choose?
  - what is the cost? (we prefer the full version)

- the lavaan syntax is easy to learn
- learning an R package will increase general R skills
- the 'mimic' option in lavaan makes a smooth transition possible from lavaan to one of the major commercial programs (and back)
- share your teaching materials:

  http://lavaan.ugent.be/resources/teaching.html/

lavaan is for applied researchers

- what matters most to many applied researchers:
  - the software is easy and intuitive to use
  - the software has all the features they want
  - results of lavaan are very close, if not identical, to those reported by their current commercial program
- a 37-pages tutorial should get you started right away

  http://lavaan.ugent.be/tutorial/

- you will be able to replicate your results (say, after ten years)

  http://lavaan.ugent.be/history/

features of lavaan

- **lavaan** is well-tested
- user-friendly fitting functions (cfa, sem, growth)
- power-user fitting function (lavaan)
- support for non-normal continuous data:
  - robust standard errors, Satorra-Bentler correction, ADF estimation, bootstrapping
- support for categorical (binary/ordinal) data
  - **lavaan** has implemented the three-stage WLS approach as developed by Bengt Muthén (1984); including robust variants (aka WLSMV)
- full support for missing data (fiml), meanstructures, and multiple groups
- linear and non-linear equality and inequality constraints

unique features

- default model specification: lavaan model syntax
  - mplus2lavaan (Michael Hallquist, included in lavaan 0.5-13)
  - lisrel2lavaan (Corbin Quick, included in the semTools package)
  - graphical (via Onyx)
  - ...
- mimic the (numerical) results of commercial packages:
  - `mimic="Mplus"`
  - `mimic="EQS"`
- new technical features:
  - informative hypothesis testing (Leonard Vanbrabant)
  - pairwise ML for binary/ordinal data (Myrsini Katsikatsou)
  - fraction of missing information (Mijke Rhemtulla)
  - ...

features NOT in lavaan (yet)

- multilevel SEM
- mixture (latent class) SEM
- ...

features we are working on

- Bayesian SEM (BUGS interface, stan interface, native)
- small-sample corrections
- causal inference
- standard errors for standardized parameters
- ML estimation for categorical data (IRT)
- ...
- better (technical) documentation

the lavaan ecosystem

- **lavaan.survey** (Daniel Oberski)

  survey weights, clustering, strata, and finite sampling corrections in SEM (including support for replicate weights, eg. PISA)

- **Onyx** (Timo von Oertzen, Andreas M. Brandmaier, Siny Tsang)

  interactive graphical interface for SEM (written in Java)

- **semTools** (Sunthud Pornprasertmanit and many others)

  collection of useful functions for SEM

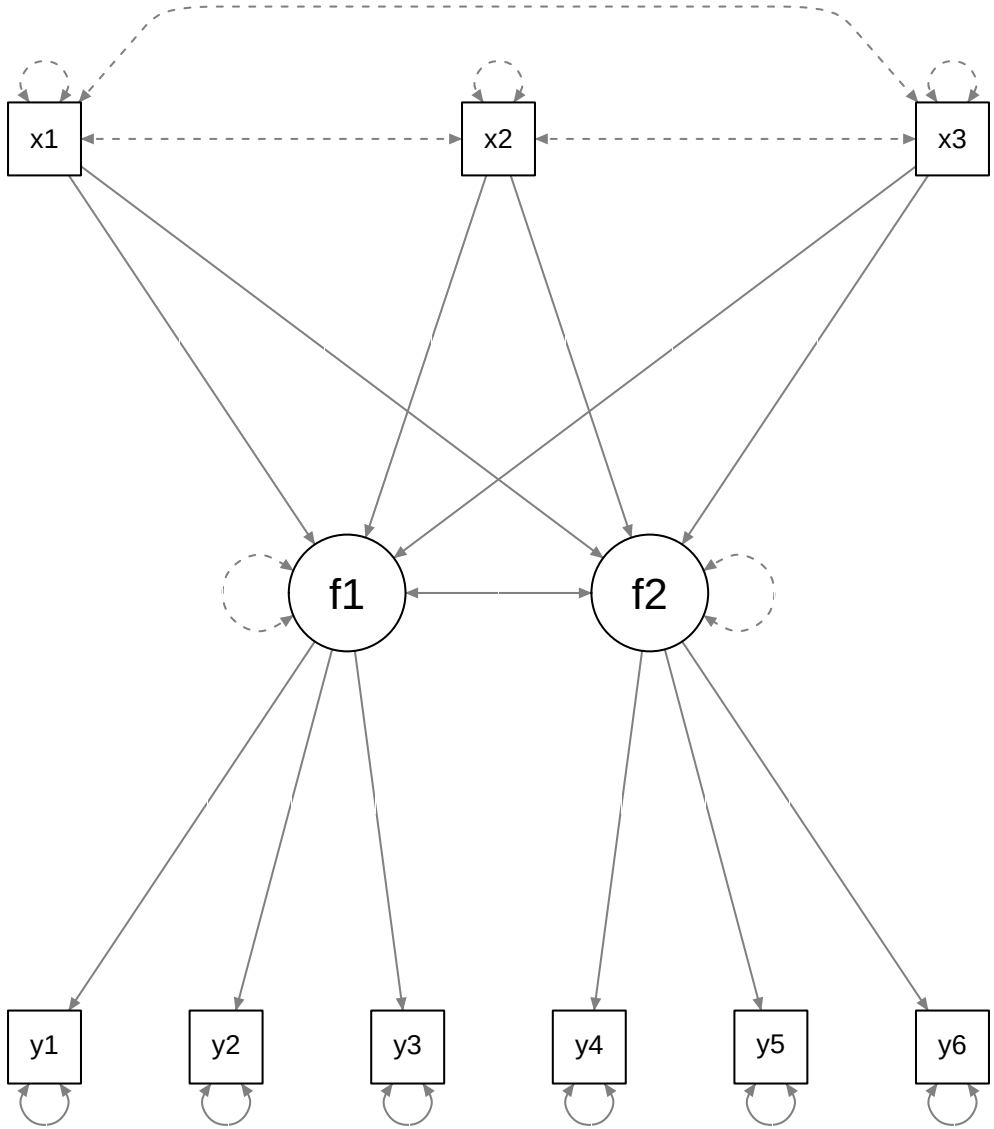
- **simsem** (Sunthud Pornprasertmanit and many others)

  simulation of SEM models

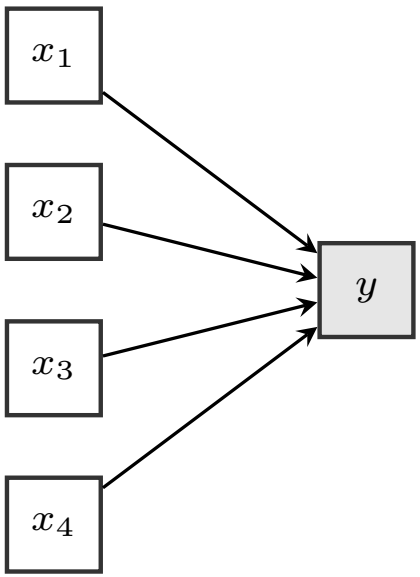
- **semPlot** (Sacha Epskamp)

  visualizations of SEM models

semPlot



a simple regression analysis in R




```
# read in your data
myData <- read.csv("c:/temp/myData.csv")

# fit model using lm
fit <- lm(formula = y ~ x1 + x2 + x3 + x4,
          data    = myData)

# show results
summary(fit)
```

The standard linear model:

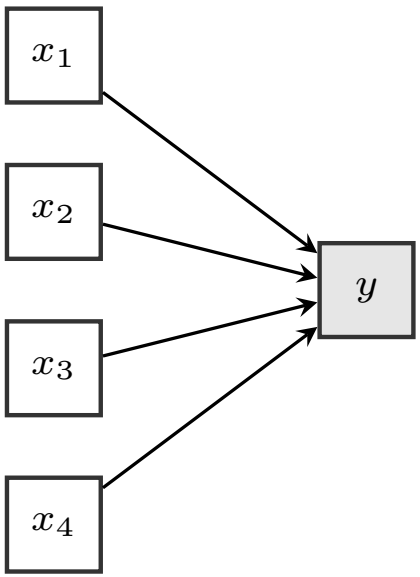
- $y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \beta_4 x_{i4} + \epsilon_i \quad (i = 1, 2, \ldots, n)$

lm() output artificial data (N=100)

```
Call:
lm(formula = y ~ x1 + x2 + x3 + x4, data = myData)

Residuals:
     Min       1Q   Median       3Q      Max
-102.372  -29.458   -3.658   27.275  148.404

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  97.7210     4.7200  20.704   <2e-16 ***
x1            5.7733     0.5238  11.022   <2e-16 ***
x2           -1.3214     0.4917  -2.688   0.0085 **
x3            1.1350     0.4575   2.481   0.0149 *
x4            0.2707     0.4779   0.566   0.5724
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 46.74 on 95 degrees of freedom
Multiple R-squared: 0.5911, Adjusted R-squared: 0.5738
F-statistic: 34.33 on 4 and 95 DF,  p-value: < 2.2e-16
```

```
# create artificial data
set.seed(1)
x1 <- rnorm(100) * 10; x2 <- rnorm(100) * 10
x3 <- rnorm(100) * 10;  x4 <- rnorm(100) * 10
y <- 100 + 5*x1 + (-2)*x2 + 1*x3 + 0.1*x4 + rnorm(100, sd=40)
myData <- data.frame(y,x1,x2,x3,x4)
```

the lavaan model syntax – a simple regression




```
library(lavaan)
myData <- read.csv("c:/temp/myData.csv")

myModel <- ' y ~ x1 + x2 + x3 + x4 '

# fit model
fit <- sem(model = myModel,
           data  = myData)

# show results
summary(fit)
```

- to 'see' the intercept, use either

```
fit <- sem(model=myModel, data=myData, meanstructure=TRUE)
```

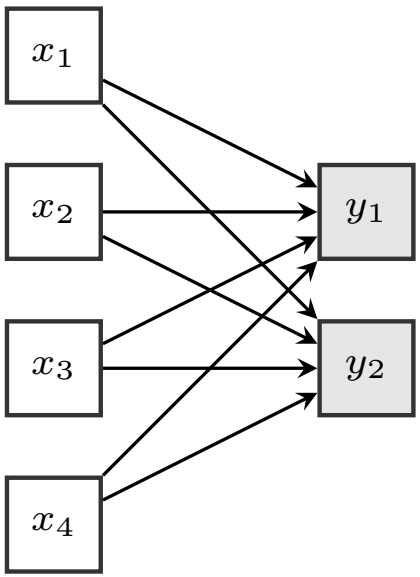
or include it explicitly in the syntax:

```
myModel <- ' y ~ 1 + x1 + x2 + x3 + x4 '
```

output (artificial data, N=100)

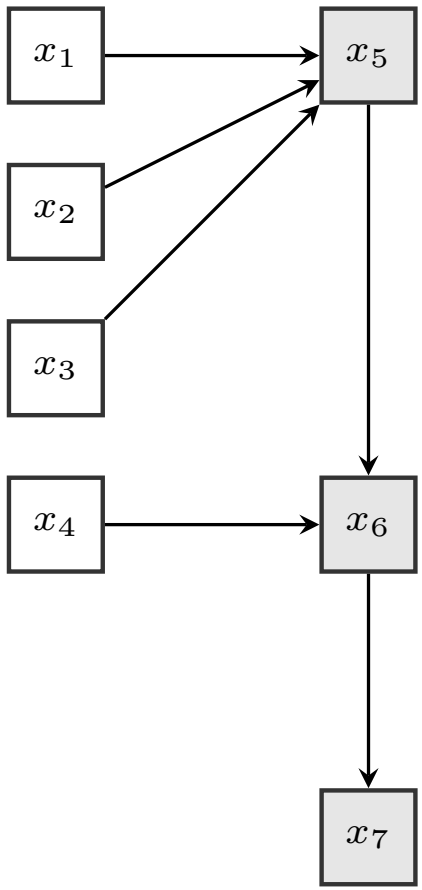
```
lavaan (0.5-13) converged normally after   1 iterations

  Number of observations                           100

  Estimator                                         ML
  Minimum Function Test Statistic                0.000
  Degrees of freedom                                 0
  P-value (Chi-square)                           1.000

Parameter estimates:

  Information                                 Expected
  Standard Errors                             Standard

                   Estimate  Std.err  Z-value  P(>|z|)
Regressions:
  y ~
    x1                5.773    0.511   11.309    0.000
    x2               -1.321    0.479   -2.757    0.006
    x3                1.135    0.446    2.545    0.011
    x4                0.271    0.466    0.581    0.561

Variances:
    y              2075.100  293.463
```

the lavaan model syntax – multivariate regression



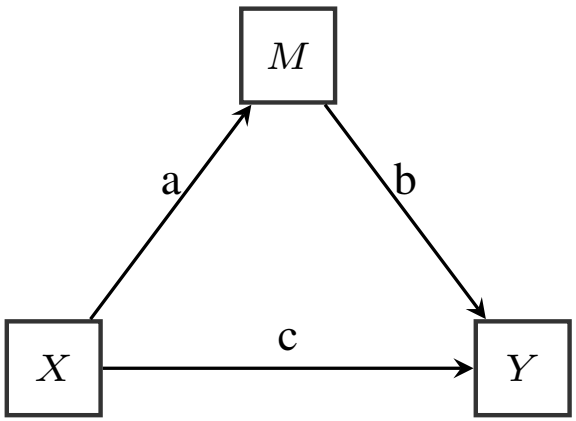

```
myModel <- ' y1 ~ x1 + x2 + x3 + x4
             y2 ~ x1 + x2 + x3 + x4 '
```

the lavaan model syntax – path analysis




```
myModel <- ' x5 ~ x1 + x2 + x3
             x6 ~ x4 + x5
             x7 ~ x6           '
```

the lavaan model syntax – mediation analysis

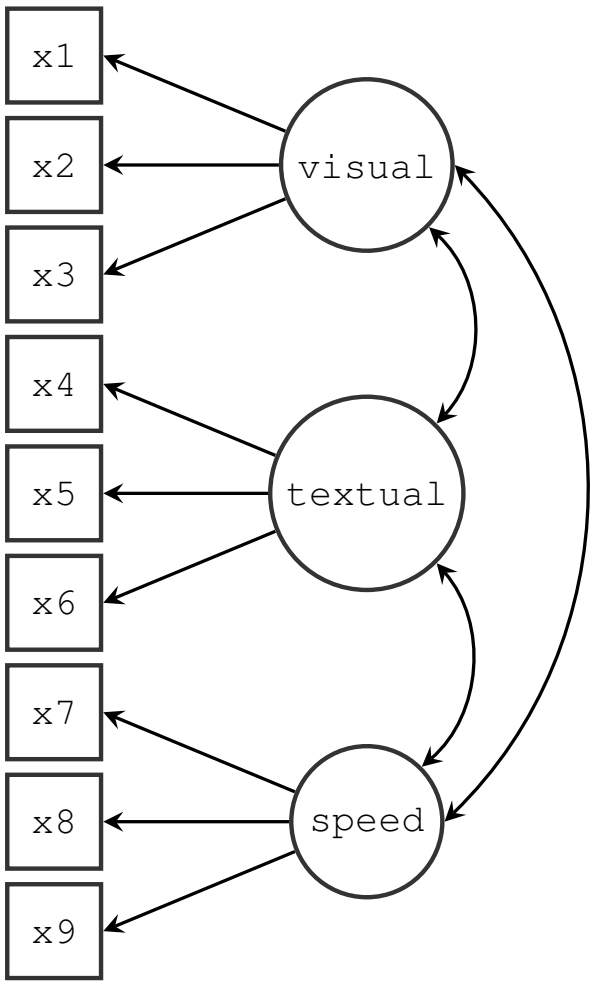



```
model <- '
           Y ~ b*M + c*X
           M ~ a*X

           indirect := a*b
           total    := c + (a*b)
         '

fit <- sem(model,
           data = myData,
           se = "bootstrap")
summary(fit)
```

output

```
...

Parameter estimates:

  Information                                 Observed
  Standard Errors                           Bootstrap
  Number of requested bootstrap draws            1000
  Number of successful bootstrap draws           1000

                   Estimate  Std.err  Z-value  P(>|z|)
Regressions:
  Y ~
    M         (b)     0.597    0.098    6.068    0.000
    X         (c)     2.594    1.210    2.145    0.032
  M ~
    X         (a)     2.739    0.999    2.741    0.006

Variances:
    Y                108.700   17.747
    M                105.408   16.556

Defined parameters:
    indirect           1.636    0.645    2.535    0.011
    total              4.230    1.383    3.059    0.002
```

the lavaan model syntax – using cfa() or sem()




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

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

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

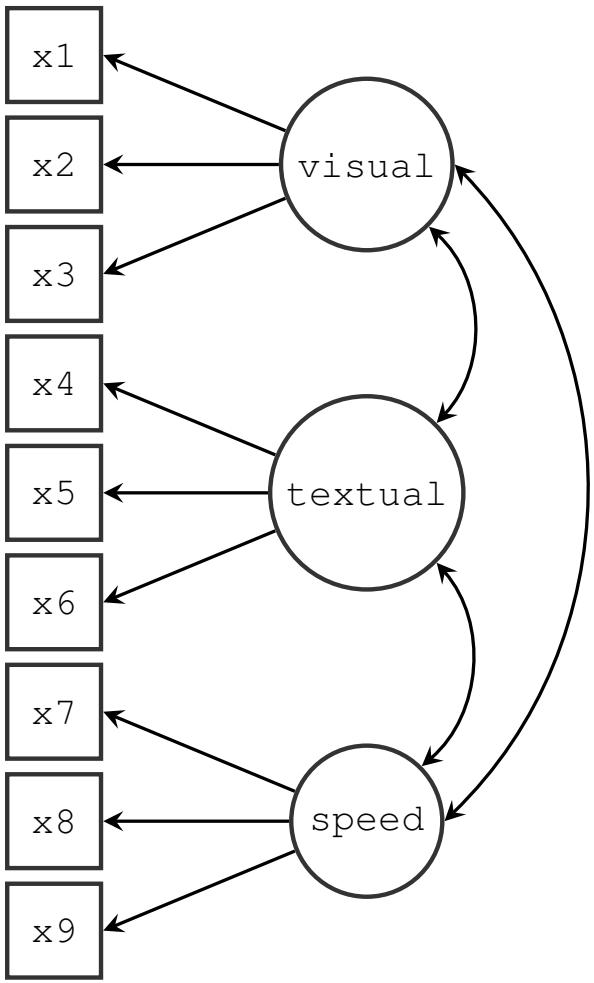
output

```
lavaan (0.5-13) converged normally after  35 iterations

  Number of observations                           301

  Estimator                                         ML
  Minimum Function Test Statistic               85.306
  Degrees of freedom                                24
  P-value (Chi-square)                           0.000

Model test baseline model:

  Minimum Function Test Statistic              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
```

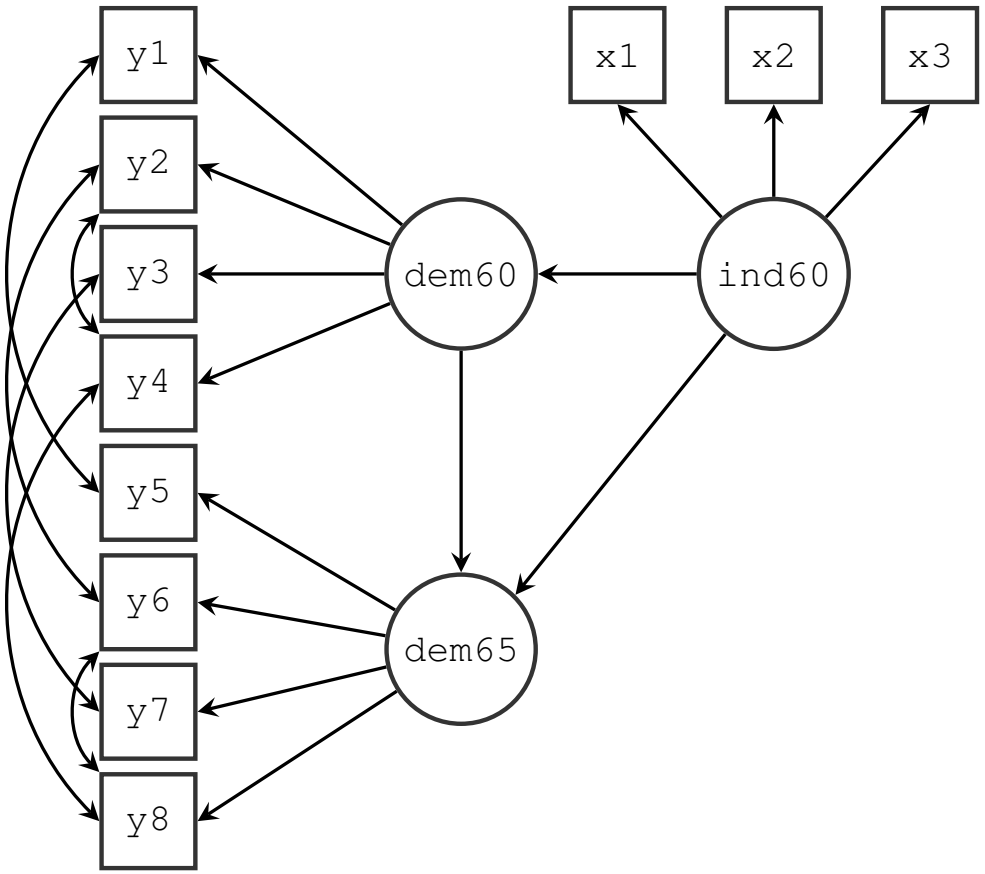
```
    x6                0.926    0.055   16.703    0.000    0.917    0.838
  speed =~
    x7                1.000                               0.619    0.570
    x8                1.180    0.165    7.152    0.000    0.731    0.723
    x9                1.082    0.151    7.155    0.000    0.670    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                      0.549    0.404
    x2                1.134    0.102                      1.134    0.821
    x3                0.844    0.091                      0.844    0.662
    x4                0.371    0.048                      0.371    0.275
    x5                0.446    0.058                      0.446    0.269
    x6                0.356    0.043                      0.356    0.298
    x7                0.799    0.081                      0.799    0.676
    x8                0.488    0.074                      0.488    0.477
    x9                0.566    0.071                      0.566    0.558
    visual            0.809    0.145                      1.000    1.000
    textual           0.979    0.112                      1.000    1.000
    speed             0.384    0.086                      1.000    1.000
```

the lavaan model syntax – using lavaan()




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

  # factor (co)variances
    visual  ~~ visual; visual  ~~ textual
    visual  ~~ speed;  textual ~~ textual
    textual ~~ speed;  speed   ~~ speed

  # residual variances
    x1 ~~ x1; x2 ~~ x2; x3 ~~ x3
    x4 ~~ x4; x5 ~~ x5; x6 ~~ x6
    x7 ~~ x7; x8 ~~ x8; x9 ~~ x9
'

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

lavaan model syntax: full sem




```
myModel <- '

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

# regressions
dem60 ~ ind60
dem65 ~ ind60 + dem60

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

'

fit <- sem(model = myModel,
           data  = ...)
```

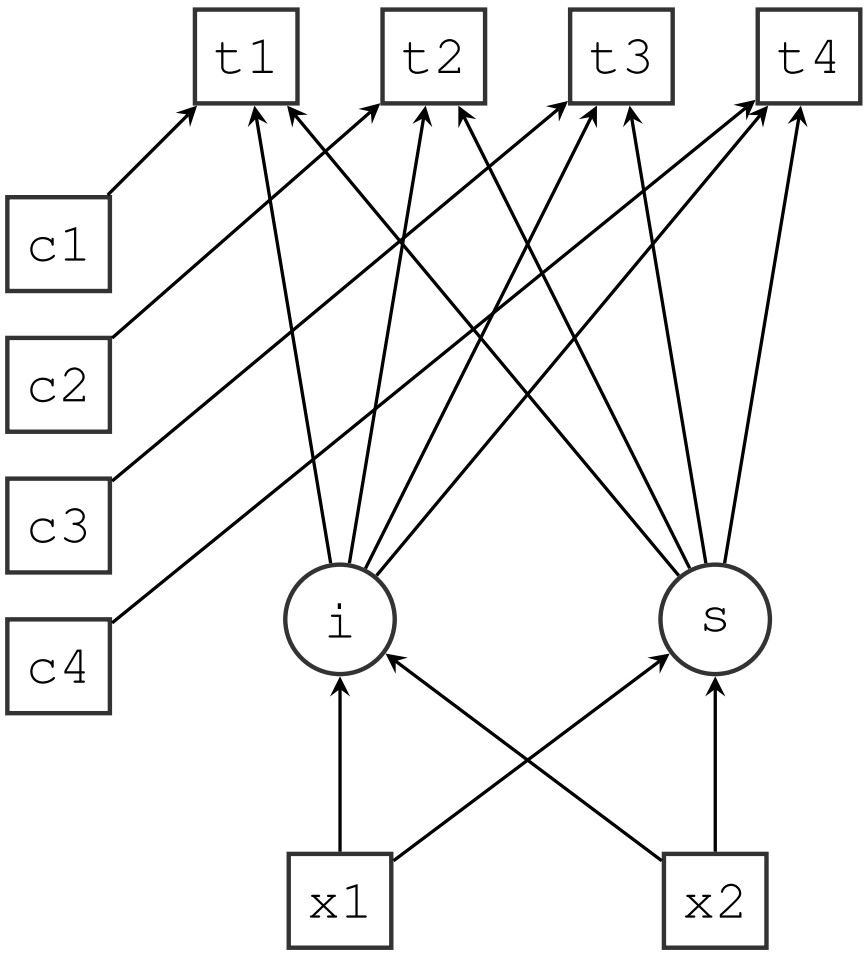
output

```
lavaan (0.5-13) converged normally after  61 iterations

  Number of observations                            75

  Estimator                                         ML
  Minimum Function Test Statistic                40.179
  Degrees of freedom                                38
  P-value (Chi-square)                           0.374

Parameter estimates:

  Information                                 Expected
  Standard Errors                             Standard

                   Estimate  Std.err  Z-value  P(>|z|)
Latent variables:
  ind60 =~
    x1                1.000
    x2                2.180    0.138   15.751    0.000
    x3                1.818    0.152   11.971    0.000
  dem60 =~
    y1                1.000
    y2         (a)    1.191    0.139    8.551    0.000
    y3         (b)    1.175    0.120    9.755    0.000
    y4         (c)    1.251    0.117   10.712    0.000
  dem65 =~
```

```
    y5                1.000
    y6         (a)    1.191    0.139    8.551    0.000
    y7         (b)    1.175    0.120    9.755    0.000
    y8         (c)    1.251    0.117   10.712    0.000

Regressions:
  dem60 ~
    ind60             1.471    0.392    3.750    0.000
  dem65 ~
    ind60             0.600    0.226    2.660    0.008
    dem60             0.865    0.075   11.554    0.000

Covariances:
  y1 ~~
    y5                0.583    0.356    1.637    0.102
  y2 ~~
    y4                1.440    0.689    2.092    0.036
    y6                2.183    0.737    2.960    0.003
  y3 ~~
    y7                0.712    0.611    1.165    0.244
  y4 ~~
    y8                0.363    0.444    0.817    0.414
  y6 ~~
    y8                1.372    0.577    2.378    0.017

Variances:
    x1                0.081    0.019
    x2                0.120    0.070
```

```
x3                0.467    0.090
y1                1.855    0.433
y2                7.581    1.366
y3                4.956    0.956
y4                3.225    0.723
y5                2.313    0.479
y6                4.968    0.921
y7                3.560    0.710
y8                3.308    0.704
ind60             0.449    0.087
dem60             3.875    0.866
dem65             0.164    0.227
```

a simple growth curve model with time-varying covariates




```
model <- '
  # random intercept and slope
    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)
```

further syntax

- fixing parameters, and overriding auto-fixed parameters

```
HS.model.bis <- ' visual  =~ NA*x1 + x2 + x3
                  textual =~ NA*x4 + x5 + x6
                  speed   =~ NA*x7 + x8 + x9
                  visual  ~~ 1*visual
                  textual ~~ 1*textual
                  speed   ~~ 1*speed
                '
```

- linear and nonlinear equality and inequality constraints

```
model.constr <- ' # model with labeled parameters
                    y ~ b1*x1 + b2*x2 + b3*x3

                  # constraints
                    b1 == (b2 + b3)^2
                    b1 > exp(b2 + b3) '
```

- several modifiers (eg. fix and label)

```
myModel <- ' y ~ 0.5*x1 + x2 + x3 + b1*x1 '
```

user-friendly fitting functions

- `cfa()` for confirmatory factor analysis
- `sem()` for path analysis and SEM
- `growth()` for growth curve modeling

arguments of the cfa() and sem() fitting functions

```
sem(model = NULL, meanstructure = "default", fixed.x = "default",
    orthogonal = FALSE, std.lv = FALSE, data = NULL, std.ov = FALSE,
    missing = "default", sample.cov = NULL, sample.mean = NULL,
    sample.nobs = NULL, group = NULL, group.equal = "",
    group.partial = "", constraints = '', estimator = "default",
    likelihood = "default", information = "default", se = "default",
    test = "default", bootstrap = 1000L,
    mimic = "default", representation = "default",
    do.fit = TRUE, control = list(), start = "default",
    verbose = FALSE, warn = TRUE, debug = FALSE)
```

power-user fitting functions

- `lavaan()` by default, no model parameters are added automagically to the parameter table
- several `auto.*` arguments are available to
  - automatically add a set of parameters (e.g. all (residual) variances)
  - take actions to make the model identifiable (e.g. set the metric of the latent variables)

example using lavaan with an auto.* argument

```
HS.model.mixed <- ' # latent variables
                      visual  =~ 1*x1 + x2 + x3
                      textual =~ 1*x4 + x5 + x6
                      speed   =~ 1*x7 + x8 + x9
                    # factor covariances
                      visual  ~~ textual + speed
                      textual ~~ speed
                  '
fit <- lavaan(HS.model.mixed, data=HolzingerSwineford1939,
              auto.var=TRUE)
```

auto.* flags

keyword	operator	parameter set
`auto.var`	`~~`	(residual) variances observed and latent variables
`auto.cov.y`	`~~`	(residual) covariances observed and latent endogenous variables
`auto.cov.lv.x`	`~~`	covariances among exogenous latent variables

keyword	default	action
`auto.fix.first`	TRUE	fix the factor loading of the first indicator to 1
`auto.fix.single`	TRUE	fix the residual variance of a single indicator to 1
`int.ov.free`	TRUE	freely estimate the intercepts of the observed variables (only if a mean structure is included)
`int.lv.free`	FALSE	freely estimate the intercepts of the latent variables (only if a mean structure is included)

extractor functions: inspecting fitted models

Method	Description
`summary()`	print a long summary of the model results
`show()`	print a short summary of the model results
`coef()`	returns the estimates of the free parameters in the model as a named numeric vector
`fitted()`	returns the implied moments (covariance matrix and mean vector) of the model
`resid()`	returns the raw, normalized or standardized residuals (difference between implied and observed moments)
`vcov()`	returns the covariance matrix of the estimated parameters
`predict()`	compute factor scores
`logLik()`	returns the log-likelihood of the fitted model (if maximum likelihood estimation was used)
`AIC()`, `BIC()`	compute information criteria (if maximum likelihood estimation was used)
`update()`	update a fitted lavaan object
`inspect()`	peek into the internal representation of the model; by default, it returns a list of model matrices counting the free parameters in the model; can also be used to extract starting values, gradient values, and much more

other functions

Function	Description
`lavaanify()`	converts a lavaan model syntax to a parameter table
`parameterTable()`	returns the parameter table
`parameterEstimates()`	returns the parameter estimates, including confidence intervals, as a data frame
`standardizedSolution()`	returns one of three types of standardized parameter estimates, as a data frame
`modindices()`	computes modification indices and expected parameter changes
`bootstrapLavaan()`	bootstrap any arbitrary statistic that can be extracted from a fitted lavaan object
`bootstrapLRT()`	bootstrap a chi-square difference test for comparing to alternative models

extractor examples (1)

```
> 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

> predict(fit)
      visual textul  speed
 [1,] -0.818 -0.138  0.062
 [2,]  0.050 -1.013  0.625
 [3,] -0.761 -1.872 -0.841
 [4,]  0.419  0.018 -0.271
 [5,] -0.416 -0.122  0.194
 ...
```

extractor examples (2)

```
> inspect(fit)  # matrix representation
$lambda
   visual textul speed
x1      0       0     0
x2      1       0     0
x3      2       0     0
x4      0       0     0
x5      0       3     0
x6      0       4     0
x7      0       0     0
x8      0       0     5
x9      0       0     6

$theta
   x1 x2 x3 x4 x5 x6 x7 x8 x9
x1  7
x2  0  8
x3  0  0  9
x4  0  0  0 10
x5  0  0  0  0 11
x6  0  0  0  0  0 12
x7  0  0  0  0  0  0 13
x8  0  0  0  0  0  0  0 14
x9  0  0  0  0  0  0  0  0 15

$psi
```

```
        visual textul speed
visual  16
textual 19     17
speed   20     21     18

> inspect(fit, "sampstat")
$cov
   x1     x2     x3     x4     x5     x6     x7     x8     x9
x1  1.358
x2  0.407  1.382
x3  0.580  0.451  1.275
x4  0.505  0.209  0.208  1.351
x5  0.441  0.211  0.112  1.098  1.660
x6  0.455  0.248  0.244  0.896  1.015  1.196
x7  0.085 -0.097  0.088  0.220  0.143  0.144  1.183
x8  0.264  0.110  0.212  0.126  0.181  0.165  0.535  1.022
x9  0.458  0.244  0.374  0.243  0.295  0.236  0.373  0.457  1.015

$mean
   x1    x2    x3    x4    x5    x6    x7    x8    x9
4.936 6.088 2.250 3.061 4.341 2.186 4.186 5.527 5.374
```

fitMeasures

```
> fitMeasures(fit)
                fmin               chisq                  df
               0.142              85.306              24.000
              pvalue       baseline.chisq          baseline.df
               0.000             918.852              36.000
      baseline.pvalue                 cfi                 tli
               0.000               0.931               0.896
                nnfi                 rfi                 nfi
               0.896               0.861               0.907
                pnfi                 ifi                 rni
               0.605               0.931               0.931
                logl        unrestricted.logl                npar
            -3737.745           -3695.092              21.000
                 aic                 bic              ntotal
            7517.490            7595.339             301.000
                bic2               rmsea      rmsea.ci.lower
            7528.739               0.092               0.071
      rmsea.ci.upper      rmsea.pvalue                 rmr
               0.114               0.001               0.082
          rmr_nomean                srmr         srmr_nomean
               0.082               0.065               0.065
               cn_05               cn_01                 gfi
             129.490             152.654               0.943
                agfi                pgfi                 mfi
               0.894               0.503               0.903
                ecvi
               0.423
```

parameterTable

```
> parameterTable(fit)
   id     lhs op     rhs user group free ustart exo label eq.id unco
1   1  visual  =~      x1    1     1    0      1   0             0    0
2   2  visual  =~      x2    1     1    1     NA   0             0    1
3   3  visual  =~      x3    1     1    2     NA   0             0    2
4   4 textual  =~      x4    1     1    0      1   0             0    0
5   5 textual  =~      x5    1     1    3     NA   0             0    3
6   6 textual  =~      x6    1     1    4     NA   0             0    4
7   7   speed  =~      x7    1     1    0      1   0             0    0
8   8   speed  =~      x8    1     1    5     NA   0             0    5
9   9   speed  =~      x9    1     1    6     NA   0             0    6
10 10      x1  ~~      x1    0     1    7     NA   0             0    7
11 11      x2  ~~      x2    0     1    8     NA   0             0    8
12 12      x3  ~~      x3    0     1    9     NA   0             0    9
13 13      x4  ~~      x4    0     1   10     NA   0             0   10
14 14      x5  ~~      x5    0     1   11     NA   0             0   11
15 15      x6  ~~      x6    0     1   12     NA   0             0   12
16 16      x7  ~~      x7    0     1   13     NA   0             0   13
17 17      x8  ~~      x8    0     1   14     NA   0             0   14
18 18      x9  ~~      x9    0     1   15     NA   0             0   15
19 19  visual  ~~  visual    0     1   16     NA   0             0   16
20 20 textual  ~~ textual    0     1   17     NA   0             0   17
21 21   speed  ~~   speed    0     1   18     NA   0             0   18
22 22  visual  ~~ textual    0     1   19     NA   0             0   19
23 23  visual  ~~   speed    0     1   20     NA   0             0   20
24 24 textual  ~~   speed    0     1   21     NA   0             0   21
```

parameterEstimates

```
> parameterEstimates(fit)
       lhs op     rhs   est    se      z pvalue ci.lower ci.upper
1   visual =~      x1 1.000 0.000     NA     NA    1.000    1.000
2   visual =~      x2 0.554 0.100  5.554      0    0.358    0.749
3   visual =~      x3 0.729 0.109  6.685      0    0.516    0.943
4  textual =~      x4 1.000 0.000     NA     NA    1.000    1.000
5  textual =~      x5 1.113 0.065 17.014      0    0.985    1.241
6  textual =~      x6 0.926 0.055 16.703      0    0.817    1.035
7    speed =~      x7 1.000 0.000     NA     NA    1.000    1.000
8    speed =~      x8 1.180 0.165  7.152      0    0.857    1.503
9    speed =~      x9 1.082 0.151  7.155      0    0.785    1.378
10      x1 ~~      x1 0.549 0.114  4.833      0    0.326    0.772
11      x2 ~~      x2 1.134 0.102 11.146      0    0.934    1.333
12      x3 ~~      x3 0.844 0.091  9.317      0    0.667    1.022
13      x4 ~~      x4 0.371 0.048  7.779      0    0.278    0.465
14      x5 ~~      x5 0.446 0.058  7.642      0    0.332    0.561
15      x6 ~~      x6 0.356 0.043  8.277      0    0.272    0.441
16      x7 ~~      x7 0.799 0.081  9.823      0    0.640    0.959
17      x8 ~~      x8 0.488 0.074  6.573      0    0.342    0.633
18      x9 ~~      x9 0.566 0.071  8.003      0    0.427    0.705
19  visual ~~  visual 0.809 0.145  5.564      0    0.524    1.094
20 textual ~~ textual 0.979 0.112  8.737      0    0.760    1.199
21   speed ~~   speed 0.384 0.086  4.451      0    0.215    0.553
22  visual ~~ textual 0.408 0.074  5.552      0    0.264    0.552
23  visual ~~   speed 0.262 0.056  4.660      0    0.152    0.373
24 textual ~~   speed 0.173 0.049  3.518      0    0.077    0.270
```

varTable

```
> varTable(fit)
  name idx nobs    type exo user  mean   var nlev lnam
1   x1   7  301 numeric   0    0 4.936 1.363    0
2   x2   8  301 numeric   0    0 6.088 1.386    0
3   x3   9  301 numeric   0    0 2.250 1.279    0
4   x4  10  301 numeric   0    0 3.061 1.355    0
5   x5  11  301 numeric   0    0 4.341 1.665    0
6   x6  12  301 numeric   0    0 2.186 1.200    0
7   x7  13  301 numeric   0    0 4.186 1.187    0
8   x8  14  301 numeric   0    0 5.527 1.025    0
9   x9  15  301 numeric   0    0 5.374 1.018    0
```

modification indices

```
> subset(modindices(fit), mi > 5)
       lhs op rhs     mi    epc sepc.lv sepc.all sepc.nox
1   visual =~  x5  7.441 -0.210  -0.189   -0.147   -0.147
2   visual =~  x7 18.631 -0.422  -0.380   -0.349   -0.349
3   visual =~  x9 36.411  0.577   0.519    0.515    0.515
4  textual =~  x1  8.903  0.350   0.347    0.297    0.297
5  textual =~  x3  9.151 -0.272  -0.269   -0.238   -0.238
6       x1 ~~  x7  5.420 -0.129  -0.129   -0.102   -0.102
7       x1 ~~  x9  7.335  0.138   0.138    0.117    0.117
8       x2 ~~  x3  8.532  0.218   0.218    0.164    0.164
9       x2 ~~  x7  8.918 -0.183  -0.183   -0.143   -0.143
10      x3 ~~  x5  7.858 -0.130  -0.130   -0.089   -0.089
11      x4 ~~  x6  6.220 -0.235  -0.235   -0.185   -0.185
12      x4 ~~  x7  5.920  0.098   0.098    0.078    0.078
13      x7 ~~  x8 34.145  0.536   0.536    0.488    0.488
14      x7 ~~  x9  5.183 -0.187  -0.187   -0.170   -0.170
15      x8 ~~  x9 14.946 -0.423  -0.423   -0.415   -0.415
```

bootstrapLavaan

```
fit <- cfa(HS.model, data=HolzingerSwineford1939, se="none")

# get the test statistic for the original sample

T.orig <- fitMeasures(fit, "chisq")

# bootstrap to get bootstrap test statistics
# we only generate 10 bootstrap sample in this example; in practice
# you may wish to use a much higher number

T.boot <- bootstrapLavaan(fit,
                          R = 10,
                          type = "bollen.stine",
                          FUN = fitMeasures,
                          fit.measures = "chisq")

# compute a bootstrap based p-value

pvalue.boot <- length(which(T.boot > T.orig))/length(T.boot)
```

testing for measurement invariance

```
# model 1: configural invariance
fit1 <- cfa(HS.model, data=HolzingerSwineford1939, group="school")

# model 2: weak invariance
fit2 <- cfa(HS.model, data=HolzingerSwineford1939, group="school",
            group.equal="loadings")

# model 3: strong invariance
fit3 <- cfa(HS.model, data=HolzingerSwineford1939, group="school",
            group.equal=c("loadings", "intercepts"))
```

comparing two (nested) models: the anova() function

```
anova(fit1, fit2)

Chi Square Difference Test

     Df    AIC    BIC  Chisq Chisq diff Df diff Pr(>Chisq)
fit1 48 7484.4 7706.8 115.85
fit2 54 7480.6 7680.8 124.04     8.1922       6     0.2244
```

- this also works correctly if the chi-square test statistics are scaled (robust)

measurement invariance tests – all together

```
> library(semTools)
> measurementInvariance(HS.model, data=HolzingerSwineford1939,
                        group="school", strict=FALSE)

Measurement invariance tests:

Model 1: configural invariance:
   chisq       df   pvalue      cfi    rmsea      bic
 115.851   48.000    0.000    0.923    0.097 7706.822

Model 2: weak invariance (equal loadings):
   chisq       df   pvalue      cfi    rmsea      bic
 124.044   54.000    0.000    0.921    0.093 7680.771

[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    0.840    0.122 7709.969

[Model 1 versus model 4]
  delta.chisq      delta.df delta.p.value     delta.cfi
       88.754        15.000         0.000         0.083

[Model 3 versus model 4]
  delta.chisq      delta.df delta.p.value     delta.cfi
       40.502         3.000         0.000         0.042
```

missing data: fiml

```
fit <- cfa(HS.model,
           data = HolzingerSwineford1939,
           missing = "ml")
```

alternative estimators

```
fit <- cfa(HS.model,
           data = HolzingerSwineford1939,
           estimator = "GLS")
```

robust standard errors

```
fit <- cfa(HS.model,
           data = HolzingerSwineford1939,
           se = "robust")
```

Satorra-Bentler scaled test statistic

```
fit <- cfa(HS.model,
           data = HolzingerSwineford1939,
           test = "Satorra-Bentler")
```

shortcut: robust standard errors and scaled test statistic

```
> fit <- cfa(HS.model,
             data = HolzingerSwineford1939,
             estimator = "MLM")
> summary(fit, fit.measures=TRUE)
lavaan (0.5-13) converged normally after  35 iterations

  Number of observations                           301

  Estimator                                         ML      Robust
  Minimum Function Test Statistic               85.306      80.872
  Degrees of freedom                                24          24
  P-value (Chi-square)                           0.000       0.000
  Scaling correction factor                                  1.055
    for the Satorra-Bentler correction

Model test baseline model:

  Minimum Function Test Statistic              918.852     789.298
  Degrees of freedom                                36          36
  P-value                                        0.000       0.000

Full model versus baseline model:

  Comparative Fit Index (CFI)                    0.931       0.925
  Tucker-Lewis Index (TLI)                       0.896       0.887
...
```

mimic option

```
> cfa(HS.model, data = HolzingerSwineford1939,
      estimator = "MLM", mimic = "EQS")
...
  Estimator                                         ML      Robust
  Minimum Function Test Statistic               85.022      81.141
...

> cfa(HS.model, data = HolzingerSwineford1939,
      estimator = "MLM", mimic = "Mplus")

...
  Estimator                                         ML      Robust
  Minimum Function Test Statistic               85.306      81.908
...

> cfa(HS.model, data = HolzingerSwineford1939,
      estimator = "MLM", mimic = "lavaan")
...
  Estimator                                         ML      Robust
  Minimum Function Test Statistic               85.306      80.872
...
```

binary and ordered categorical data

```
# binary version of Holzinger & Swineford
HS9 <- HolzingerSwineford1939[,c("x1","x2","x3","x4","x5",
                                 "x6","x7","x8","x9")]
HSbinary <- as.data.frame( lapply(HS9, cut, 2, labels=FALSE) )

# single factor model
model <- ' visual  =~ x1 + x2 + x3
           textual =~ x4 + x5 + x6
           speed   =~ x7 + x8 + x9 '

# binary CFA
fit <- cfa(model, data=HSbinary, ordered=names(HSbinary))

# summary
summary(fit, fit.measures=TRUE)
```

output

```
lavaan (0.5-13) converged normally after  36 iterations

  Number of observations                           301

  Estimator                                       DWLS      Robust
  Minimum Function Test Statistic               30.918      38.546
  Degrees of freedom                                24          24
  P-value (Chi-square)                           0.156       0.030
  Scaling correction factor                                  0.866
  Shift parameter                                            2.861
    for simple second-order correction (Mplus variant)

Model test baseline model:

  Minimum Function Test Statistic              582.533     469.769
  Degrees of freedom                                36          36
  P-value                                        0.000       0.000

Full model versus baseline model:

  Comparative Fit Index (CFI)                    0.987       0.966
  Tucker-Lewis Index (TLI)                       0.981       0.950

Root Mean Square Error of Approximation:

  RMSEA                                          0.031       0.045
```

```
  90 Percent Confidence Interval          0.000  0.059       0.014  0.070
  P-value RMSEA <= 0.05                          0.848       0.598

Parameter estimates:

  Information                                 Expected
  Standard Errors                          Robust.sem

                   Estimate  Std.err  Z-value  P(>|z|)
Latent variables:
  visual =~
    x1                1.000
    x2                0.900    0.188    4.788    0.000
    x3                0.939    0.197    4.766    0.000
  textual =~
    x4                1.000
    x5                0.976    0.118    8.241    0.000
    x6                1.078    0.125    8.601    0.000
  speed =~
    x7                1.000
    x8                1.569    0.461    3.403    0.001
    x9                1.449    0.409    3.541    0.000

Covariances:
  visual ~~
    textual           0.303    0.061    4.981    0.000
    speed             0.132    0.049    2.700    0.007
  textual ~~
```

```
    speed             0.076    0.046    1.656    0.098

Intercepts:
    visual            0.000
    textual           0.000
    speed             0.000

Thresholds:
    x1|t1            -0.388    0.074   -5.223    0.000
    x2|t1            -0.054    0.072   -0.748    0.454
    x3|t1             0.318    0.074    4.309    0.000
    x4|t1             0.180    0.073    2.473    0.013
    x5|t1            -0.257    0.073   -3.506    0.000
    x6|t1             1.024    0.088   11.641    0.000
    x7|t1             0.231    0.073    3.162    0.002
    x8|t1             1.128    0.092   12.284    0.000
    x9|t1             0.626    0.078    8.047    0.000

Variances:
    x1                0.592
    x2                0.670
    x3                0.640
    x4                0.303
    x5                0.336
    x6                0.191
    x7                0.778
    x8                0.453
    x9                0.534
```

```
    visual             0.408    0.112
    textual            0.697    0.101
    speed              0.222    0.094
```

```
> inspect(fit, "sampstat")
$cov
   x1     x2     x3     x4     x5     x6     x7     x8     x9
x1  1.000
x2  0.284  1.000
x3  0.415  0.389  1.000
x4  0.364  0.328  0.232  1.000
x5  0.319  0.268  0.138  0.688  1.000
x6  0.422  0.322  0.206  0.720  0.761  1.000
x7 -0.048  0.061  0.041  0.200  0.023 -0.029  1.000
x8  0.159  0.105  0.439 -0.029 -0.059  0.183  0.464  1.000
x9  0.165  0.210  0.258  0.146  0.183  0.230  0.335  0.403  1.000

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

$th
 x1|t1  x2|t1  x3|t1  x4|t1  x5|t1  x6|t1  x7|t1  x8|t1  x9|t1
-0.388 -0.054  0.318  0.180 -0.257  1.024  0.231  1.128  0.626
```

the history of SEM, from a computational point of view

- several traditions in the SEM (software) world:
  - LISREL (Karl Jöreskog)
  - EQS (Peter Bentler)
  - Mplus (Bengt Muthén)
  - RAM-based approaches (AMOS, Mx, sem, OpenMx, . . .)
- superficially, all SEM software packages produce the same results
- there are some subtle (and less subtle) differences in the output
- looking deeper, there are many computational differences

some differences

- matrix representation
  - standard number of matrices: LISREL: 8; Mplus: 4, EQS: 3, RAM: 2
- optimization algorithm
  - quasi-Newton, gradient-only + quasi-Newton, Gauss-Newton, ...
- variances constrained (strictly positive) versus unrestricted
- constrained optimization algorithm
  - mostly undocumented
  - a Lagrangian-multiplier variant, simple slacks, ...
- normal likelihood versus Wishart likelihood, ML versus RLS
  - N versus $N - 1$
  - GLS-ML based chi-square test statistic influences fit measures (CFI!)

some differences (2)

- Satorra-Bentler/Yuan-Bentler scaled test statistic
  - each program seems to use a different implementation
  - often asymptotically equivalent; but large differences in small samples
- categorical data using the limited information approach
  - Muthén 1984; Jöreskog 1994; Lee, Poon, Bentler (1992)
  - many ways to compute the asymptotic covariance matrix (needed for WLS)
- naive bootstrapping, Bollen-Stine bootstrapping
  - mostly undocumented; one-iteration bootstrap?
  - Bollen-Stine with missing data

...

lavaan and the history of SEM

- lavaan tries to preserve the many differences from the several traditions
- all fitting functions in lavaan have a `mimic` argument:
  - `mimic="EQS"`, `mimic="Mplus"`
  - `mimic="LISREL"` is under development
  - this was originally intended to convince users that lavaan could produce 'identical' results as the (commercial) competition
  - it is now one of the main design goals of lavaan
  - for reproducibility, we plan to make 'mimic' version-specific: `mimic="Mplus4"`, `mimic="LISREL8"`, ...
- we need to (re)evaluate old/new/unexplored computational methods in many areas (optimization, constrained inference, Bayesian techniques, limited information estimation, ...)

version differences: Mplus 6/7 versus Mplus <6 (HSbinary example)

```
Mplus VERSION 7 (Linux)
...
MODEL FIT INFORMATION

Number of Free Parameters                       21

Chi-Square Test of Model Fit

          Value                             38.546*
          Degrees of Freedom                    24
          P-Value                           0.0305
...

Mplus VERSION 4.1
...
TESTS OF MODEL FIT

Chi-Square Test of Model Fit

          Value                             32.074*
          Degrees of Freedom                    19**
          P-Value                           0.0307
...
```

Satterthwaite versus simple second-order correction

```
> cfa(model, data=HSbinary, ordered=names(HSbinary), mimic="Mplus")

...
  Estimator                                      DWLS      Robust
  Minimum Function Test Statistic              30.918      38.546
  Degrees of freedom                               24          24
  P-value (Chi-square)                          0.156       0.030
  Scaling correction factor                                 0.866
  Shift parameter                                           2.861
    for simple second-order correction (WLSMV)

> cfa(model, data=HSbinary, ordered=names(HSbinary), mimic="Mplus",
      estimator="WLSMVS")

...
  Estimator                                      DWLS      Robust
  Minimum Function Test Statistic              30.918      32.074
  Degrees of freedom                               24          19
  P-value (Chi-square)                          0.156       0.031
  Scaling correction factor                                 0.964
    for the mean and variance adjusted correction (WLSMV)
```

future plans (1)

- lavaan should 'by default' implement best practices in all areas
  - we welcome advice from scholars
  - more simulations studies may be needed
  - 'defaults' settings should be flexible (depending on the model, depending on the data)
- lavaan should be 100% documented
  - every single number computed by lavaan should be documented
  - this includes the 'mimic' variations
  - all documentation should be available online
- lavaan should become the golden standard
  - all computations can be checked at the source code level
  - there should be a one-to-one mapping between the actual computations (at the source level) and the technical documentation

future plans (2)

- lavaan should become a computational toolbox
  - statisticians should be able to quickly (and reliably) implement their new statistical (modern) modeling ideas
  - modular, self-contained building blocks
  - application programming interface (API)
  - ...
- lavaan should be extremely fast
  - we hope to write a kernel in C++ soon
- lavaan should be available in other languages
  - python
  - Julia
  - Matlab
  - ...

future plans (3)

- lavaan should become the Rosetta stone for SEM:
  - read/parse syntax from many external programs (we already have `mplus2lavaan()` and `lisrel2lavaan()`)
  - `lavExport()`: write/export syntax to external programs (we can already write Mplus input files)
- lavaan should be able to detect potential problems in your data and/or model automatically
  - warning and error messages should be informative
  - computing on the model (as represented by the parameter table)
- lavaan version 1.0

Thank you!

(questions?)

`http://lavaan.org`