SEM with lavaan: syntax overview

Ben Kite[1], Paul Johnson[1], and others[2]

[1]Center for Research Methods and Data Analysis [2]Previous GRAs, including but not limited to Terry Jorgensen, Sunthud Pornprasertmanit, Jared Harpole

2019

KU

Outline

KU

There is no SEM in R base

- Neither R (R Core Team, 2017) nor any of the required or recommended packages distributed with it include structural equation models. But, ...
- R includes a programming language and functions with which SEM estimators have been created

SEM packages

- `sem()` in the "`car`" package by John Fox. Nearly as old as R itself
- `lavaan`, a package prepared by Yves Rosseel, who intends a function-by-function replication of the results in Mplus
- `OpenMX`, a long-standing matrix calculation framework spearheaded by Steven Bolker, which has been re-written as an R package.
- Various other R packages exist, either as wholesale replacements (`lava`) or as supplementary tools (`semPlot`).

Here we are focused on lavaan

- lavaan is the closest to a "full suite" of SEM tools needed by researchers
  - different estimation algorithms (FIML, WLSMV, etc)
- Comprehensive essay,
  Rosseel, Yves (2012). lavaan: An R Package for Structural Equation Modeling. *Journal of Statistical Software*, 48(2), 1 - 36.
  doi:http://dx.doi.org/10.18637/jss.v048.i02

KU

Use affect data

This data is about positive and negative affect. We use it in several of our training workshops

```
ddir <- "data"
fn <- "affect-2.csv"
dat <- read.csv(file.path(ddir, fn))
```

```
head(dat)
```

```
  Agency1 Agency2 Agency3 Intrin1 Intrin2 Intrin3 Extrin1 Extrin2 Extrin3 PosAFF1 PosAFF2 PosAFF3
1  3.5000  4.0000  4.0000  4.0000     4.0       4  1.0000  1.0000  1.5000  4.0000     4.0     4.0
2  2.5000  3.1667  3.0000  3.2123     2.0       3  1.8333  2.6667  1.8333  3.0000     3.5     2.5
3  1.8333  2.0000  1.5000  3.0000     3.0       2  1.0000  1.0000  1.0000  3.0184     2.5     3.0
4  2.7714  3.0602  2.3639  3.1337     4.0       3  1.0774  1.1667  1.0000  3.0000     2.5     3.0
5  3.1667  3.3333  2.8333  3.5000     4.0       4  1.8333  2.0000  1.8333  3.7804     3.5     3.0
6  2.3333  2.8333  2.3333  3.0000     2.5       3  3.0588  2.4125  2.6667  4.0000     3.0     3.0
  NegAFF1 NegAFF2 NegAFF3 Sex gender ethnicity     race
1     1.0  1.0000     1.0   1   male  Hispanic Nonwhite
2     1.5  1.6858     1.5   1   male     White    White
3     1.0  1.0000     1.0   1   male     White    White
4     2.5  2.5000     1.5   1   male     White    White
5     2.5  2.0000     3.0   1   male     White    White
6     2.0  1.5000     2.0   1   male     White    White
```

Quick Jargon review

Indicators: The observed (aka "manifest") variables

Latent variables: (aka "factors", "latent constructs" or "common factors"): unobserved variables thought to be the things we are truly interested in. We'd really like to study the relationship among them, but we are unable to do so.

factor loadings: the coefficients which indicate how tightly an indicator is linked to the latent variable.

lavaan's "model fitting" functions

- The key lavaan modeling functions are
  - `lavaan()` : a general model fitter
  - `sem()` : a convenience "wrapper" for *fitting structural equation models*
  - `cfa()` : A convenience for *confirmatory factor analysis*
  - Def. "wrapper": a function that receives the input, supplies other default parameters, and then asks other functions to do their work (in this case, `lavaan`)

Seemingly overwhelming number of arguments

- The help pages for `sem` and `cfa` list a lot of arguments

```
library(lavaan)
##Run:
?lavaan
?parameterEstimates
```

The most important arguments

- `model` : a formula (character variable representing a formula)
- `data` : name of data.frame containing variables
- `estimator` : model estimator, defaults to "ML"
- `ordered` : vector of variables to be treated as ordered-categorical

Outline

Start with R regression syntax

- A regression in R:

```
mod1 <- lm(PosAFF1 ~ Agency1 + Intrin1, data =
    dat)
summary(mod1)
```

```
Call:
lm(formula = PosAFF1 ~ Agency1 + Intrin1, data = dat)

Residuals:
     Min       1Q   Median       3Q      Max
-2.17693 -0.35887  0.05093  0.55123  1.12382

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.05775    0.18125  11.353  < 2e-16 ***
Agency1      0.18594    0.06581   2.825  0.00497 **
Intrin1      0.20778    0.04414   4.707 3.54e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6363 on 377 degrees of freedom
Multiple R-squared:  0.09673, Adjusted R-squared:  0.09194
```

Start with R regression syntax ...

```
F-statistic: 20.19 on 2 and 377 DF,  p-value: 4.693e-09
```

The syntax for the formula "PosAFF1 ~ Agency1 + Intrin1" is *in* the function call.

- We could write the formula separately, `fmla2`, then use the formula in the `lm()` function

```
fmla2 <- "PosAFF1 ~ Agency1 + Intrin1"
mod2 <- lm(fmla2, data = dat)
```

- SEM are "larger" formulas, so we take that 2-step approach. Usually.

Lavaan formula-writing tools

1. `=~` defines latent variables (measurement models)
2. `~~` sets covariances between latent variables
3. `~` is for regression, as in R regression

Will now illustrate

lavaan symbol "=∼"

- Syntax for linkage between observed "indicators" and unmeasured "latent" variables is like so:

```
cfa.mod <- '
 a_latent_variable =~ indicator1 + indicator2 +
    indicator3
 '
```

- " a_latent_variable " whatever name we want. Could be " xi_1 " or anything else.
- " indicator1 ", " indicator2 ", " indicator3 " must be a variables named in the data frame
  - Example

```
cfa.mod <- '
 Agency =~ Agency1 + Agency2 + Agency3
 '
```

lavaan symbol "=∼" ...

- Some pressure to keep variable names short, to make typing easier
- You can use single or double quotes
- Line breaks are visual enhancement, which lavaan understands correctly

lavaan symbol "=~" ...

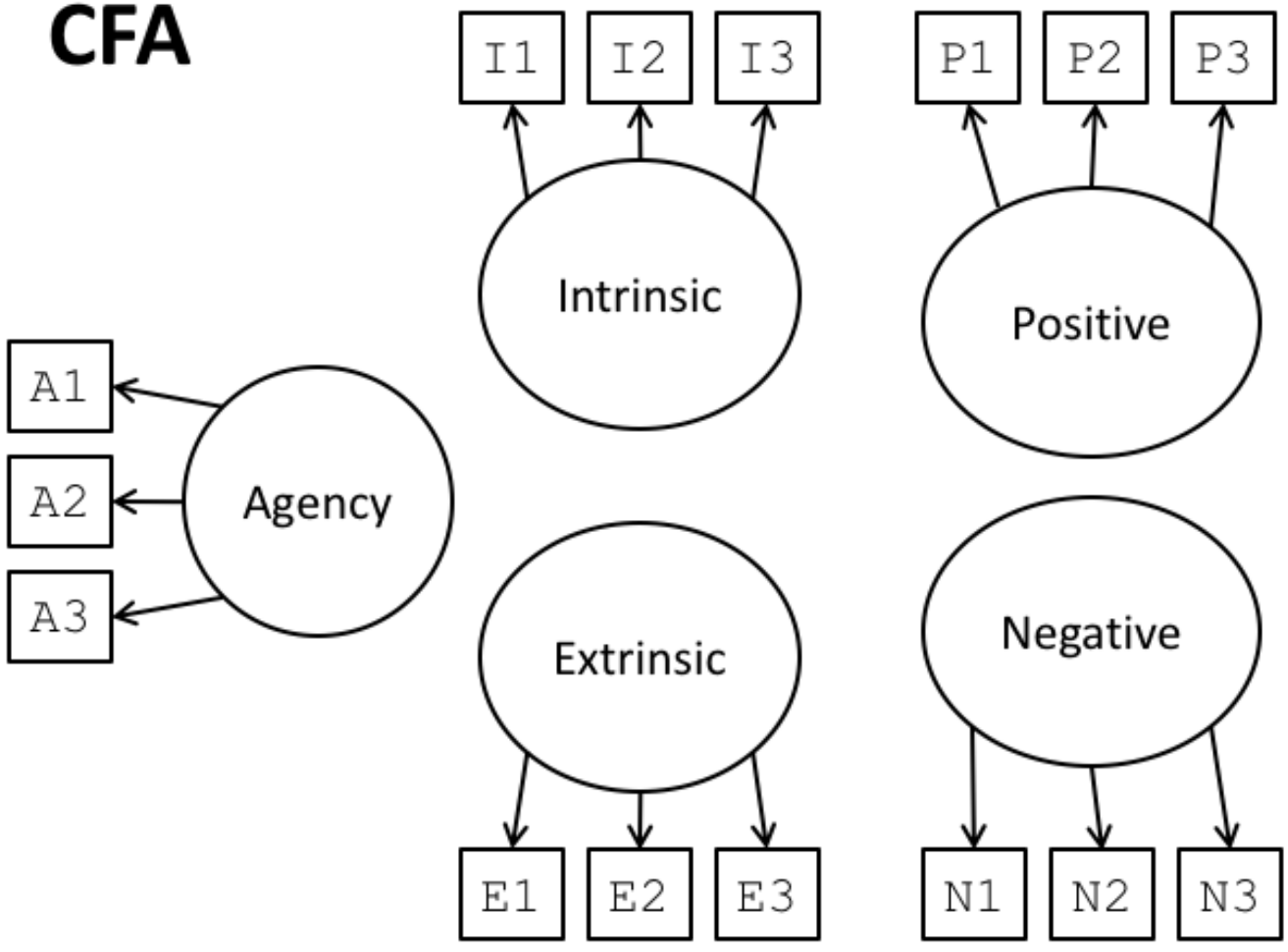
- lavaan `=~` syntax looks backwards to some. Math would have indicators on left, and the latent variable on right:

$$Agency1_i = \alpha_1 + \lambda_1 Agency_i + \varepsilon_{1i}$$
$$Agency2_i = \alpha_2 + \lambda_2 Agency_i + \varepsilon_{2i}$$
$$Agency3_i = \alpha_3 + \lambda_3 Agency_i + \varepsilon_{3i}$$

- book-keeping
  - i is a "case" index (people in survey)
  - α (alpha) is the "intercept" (often called "mean" b/c it is expected value when LV is 0)
  - λ (lambda) is the factor loading
  - ε_{ji} the error term 2 subscripts, one for the equation and one for the individual case
- Mplus, a leading commercial program, introduced that backwards syntax.

KU

Assignment: Make a lavaan formula for this



Assignment: Make a lavaan formula for this ...

See hints next slides

Assignment: Make a lavaan formula for this ...

Hint 1: You must use names of actual variables on RHS. Run "colnames(dat)" to see the column names.

Hint 2: Fill in 5 rows of formulae

```
cfa.mod <- '
Agency =~ Agency1 + Agency? + ?
? =~ ? + ? + ?
? =~ ? + ? + ?
'
```

On LHS, you can choose whatever latent variable name you want!

You should end up with something like this

```
cfa.mod <- '
 Agency =~ Agency1 + Agency2 + Agency3
 Intrinsic =~ Intrin1 + Intrin2 + Intrin3
 Extrinsic =~ Extrin1 + Extrin2 + Extrin3
 Positive =~ PosAFF1 + PosAFF2 + PosAFF3
 Negative =~ NegAFF1 + NegAFF2 + NegAFF3
 '
```

- One lesson: Pick simpler names. Wish we had them!

Could instead write out one element per indicator

```
cfa.mod <- '
 Agency =~ Agency1
 Agency =~ Agency2
 Agency =~ Agency3
 Intrinsic =~ Intrin1
 Intrinsic =~ Intrin2
 Intrinsic =~ Intrin3
 Extrinsic =~ Extrin1
 Extrinsic =~ Extrin2
 Extrinsic =~ Extrin3
 Positive =~ PosAFF1
 Positive =~ PosAFF2
 Positive =~ PosAFF3
 Negative =~ NegAFF1
 Negative =~ NegAFF2
 Negative =~ NegAFF3
 '
```

Could instead write out one element per indicator ...

- That is verbose, we don't generally write it that way
- This may be easier in very large projects where functions are writing out the formula for us.

Another lavaan symbol: Covariance "∼∼"

- To estimate the covariance between two latent variables, the symbol `~~` (that's two tilde) is used.

```
latent1 ∼∼ latent2
```

- Must use same latent variable names as are declared in the indicator part of the model.
- Some models assume covariance must be estimated, even if we don't request it (will see example)

In lavaan, $\sim$ represents a regression

The symbol `latentA ~ latentB` indicates that `latentB` predicts `latentA` in the regression sense.

We ask SEM to calculate coefficients for a regression model:

$$latentA_i = \beta_0 + \beta_1 latentB_i + v_i$$

These slope coefficients represent "directionality".

Rather than covarying, latentB predicts latentA (could be a causal relationship).

Coefficients can be named for future reference

In lavaan formulae, coefficients can be named

```
latentA ~ beta1*latentB
```

```
latentA ~ beta1 * latentB
```

Outline

KU

Example model syntax for Confirmatory Factor Analysis

```
cfa.mod <- '
 Agency =~ Agency1 + Agency2 + Agency3
 Intrinsic =~ Intrin1 + Intrin2 + Intrin3
 Extrinsic =~ Extrin1 + Extrin2 + Extrin3
 Positive =~ PosAFF1 + PosAFF2 + PosAFF3
 Negative =~ NegAFF1 + NegAFF2 + NegAFF3
 '
```

- Note, we do not include any `~~` symbols here

Estimate with cfa

```
cfa1 <- cfa(cfa.mod, data = dat)
```

- If there is no output, the model was estimated without errors
- If you did not do it yet, read " `?cfa` "
- As in other R model fitters, the `summary()` method is used to overview the results

```
summary(cfa1)
```

Estimate with cfa ...

```
lavaan 0.6-3 ended normally after 66 iterations

  Optimization method                           NLMINB
  Number of free parameters                         40

  Number of observations                           380

  Estimator                                         ML
  Model Fit Test Statistic                     106.847
  Degrees of freedom                                80
  P-value (Chi-square)                           0.024

Parameter Estimates:

  Information                                 Expected
  Information saturated (h1) model          Structured
  Standard Errors                             Standard

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)
  Agency =~
    Agency1           1.000
    Agency2           1.054    0.036   29.454    0.000
    Agency3           1.065    0.038   27.987    0.000
  Intrinsic =~
```

Estimate with cfa ...

```
    Intrin1           1.000
    Intrin2           1.075    0.097   11.043    0.000
    Intrin3           1.138    0.096   11.832    0.000
  Extrinsic =~
    Extrin1           1.000
    Extrin2           1.177    0.077   15.356    0.000
    Extrin3           1.213    0.077   15.720    0.000
  Positive =~
    PosAFF1           1.000
    PosAFF2           1.060    0.049   21.607    0.000
    PosAFF3           1.110    0.051   21.963    0.000
  Negative =~
    NegAFF1           1.000
    NegAFF2           0.923    0.038   24.210    0.000
    NegAFF3           0.944    0.037   25.639    0.000

Covariances:
                   Estimate  Std.Err  z-value  P(>|z|)
  Agency ~~
    Intrinsic         0.128    0.018    7.050    0.000
    Extrinsic         0.049    0.011    4.486    0.000
    Positive          0.080    0.016    5.152    0.000
    Negative          0.005    0.016    0.312    0.755
  Intrinsic ~~
    Extrinsic        -0.006    0.013   -0.440    0.660
    Positive          0.127    0.021    5.957    0.000
```

Estimate with cfa ...

```
    Negative          0.007    0.021    0.341    0.733
  Extrinsic ~~
    Positive         -0.006    0.013   -0.502    0.616
    Negative          0.051    0.015    3.514    0.000
  Positive ~~
    Negative         -0.026    0.020   -1.264    0.206

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .Agency1           0.048    0.005    9.649    0.000
   .Agency2           0.036    0.005    7.604    0.000
   .Agency3           0.051    0.005    9.279    0.000
   .Intrin1           0.298    0.029   10.204    0.000
   .Intrin2           0.386    0.036   10.620    0.000
   .Intrin3           0.214    0.029    7.474    0.000
   .Extrin1           0.080    0.009    8.924    0.000
   .Extrin2           0.123    0.013    9.465    0.000
   .Extrin3           0.103    0.012    8.264    0.000
   .PosAFF1           0.121    0.012    9.921    0.000
   .PosAFF2           0.104    0.012    8.615    0.000
   .PosAFF3           0.099    0.013    7.870    0.000
   .NegAFF1           0.107    0.012    9.202    0.000
   .NegAFF2           0.096    0.010    9.498    0.000
   .NegAFF3           0.067    0.009    7.369    0.000
    Agency            0.218    0.019   11.280    0.000
    Intrinsic         0.292    0.042    6.980    0.000
```

Estimate with cfa ...

```
    Extrinsic         0.150    0.017    8.803    0.000
    Positive          0.324    0.032   10.066    0.000
    Negative          0.402    0.037   10.835    0.000
```

Why did we use cfa, not lavaan, to fit that?

- Could have used the `lavaan()` function,
- But `cfa()` has customized settings
- See "`?cfa`". `cfa()` sets defaults that are customary for confirmatory factor analysis.
  - Estimate observed variable intercepts
  - set the means of the latent variables to 0
  - fix the loading for the first indicator to 1 (unless std.lv = TRUE), and
  - estimate the covariances between latent variables (even though we did not explicitly ask for them).

KU

CFA assumes you want to estimate variances among the latent variables: "~~"

- Reminder: covariance estimate is requested by `latent1 ~~ latent2`.
- We could have asked, explicitly, for covariances

```
cfa.mod2 <- '
 Agency =~ Agency1 + Agency2 + Agency3
 Intrinsic =~ Intrin1 + Intrin2 + Intrin3
 Extrinsic =~ Extrin1 + Extrin2 + Extrin3
 Positive =~ PosAFF1 + PosAFF2 + PosAFF3
 Negative =~ NegAFF1 + NegAFF2 + NegAFF3
 Agency ~~ Intrinsic
 Agency ~~ Extrinsic
 Agency ~~ Positive
 Agency ~~ Negative
 Intrinsic ~~ Extrinsic
 Intrinsic ~~ Positive
 Intrinsic ~~ Negative
```

CFA assumes you want to estimate variances among the latent variables: "∼∼" ...

```
  Extrinsic ~~ Positive
15 Extrinsic ~~ Negative
  Positive ~~ Negative
  '
```

- But cfa makes this unnecessary, we "got them for free".

The mathematical formula

- Writing out each equation for each indicator will be exhausting

$$
\begin{aligned}
Agency1_i &= \alpha_{11} + \lambda_{11} Agency_i + \varepsilon_{11i} \\
Agency2_i &= \alpha_{12} + \lambda_{21} Agency_i + \varepsilon_{12i} \\
Agency3_i &= \alpha_{13} + \lambda_{31} Agency_i + \varepsilon_{13i} \\
Extrin1_i &= \alpha_{21} + \lambda_{42} Extrinsic_i + \varepsilon_{21i} \\
Extrin2_i &= \alpha_{22} + \lambda_{52} Extrinsic_i + \varepsilon_{22i} \\
Extrin3_i &= \alpha_{23} + \lambda_{62} Extrinsic_i + \varepsilon_{23i} \\
&\text{and so forth}
\end{aligned}
$$

Now I'm thinking in Matrix terms

I'll just write out 2 LV's worth

$$\begin{bmatrix} Agency1_i \\ Agency2_i \\ Agency3_i \\ Extrin1_i \\ Extrin2_i \\ Extrin3_i \end{bmatrix} = \begin{bmatrix} \alpha_{11} \\ \alpha_{12} \\ \alpha_{13} \\ \alpha_{21} \\ \alpha_{22} \\ \alpha_{23} \end{bmatrix} + \begin{bmatrix} \lambda_{11} & 0 \\ \lambda_{21} & 0 \\ \lambda_{31} & 0 \\ 0 & \lambda_{42} \\ 0 & \lambda_{52} \\ 0 & \lambda_{62} \end{bmatrix} \begin{bmatrix} Agency_i \\ Extrinsic_i \end{bmatrix} + \begin{bmatrix} \varepsilon_{11i} \\ \varepsilon_{12i} \\ \varepsilon_{13i} \\ \varepsilon_{21i} \\ \varepsilon_{23i} \\ \varepsilon_{23i} \end{bmatrix}$$

Subscripts on loading coefficients: subscript 1 is "row", subscript 2 is "column".

Matrix view

- Try to get used to the matrix "LISREL" notation

$$\mathbf{x}_i = \boldsymbol{\alpha} + \boldsymbol{\Lambda}\boldsymbol{\xi}_i + \boldsymbol{\varepsilon}_i$$

- The latent variable vector is $\boldsymbol{\xi}_i$, as in

$$\boldsymbol{\xi}_i = \begin{bmatrix} \xi_{1i} \\ \xi_{2i} \end{bmatrix} = \begin{bmatrix} Agency_i \\ Extrinsic_i \end{bmatrix}$$

  one row per latent variable

- observed variables $\mathbf{x}_i$, all indicators "stacked" in one column

$$\mathbf{x}_i = \begin{bmatrix} Agency1_i \\ Agency2_i \\ Agency3_i \\ Extrin1_i \\ Extrin2_i \\ Extrin3_i \end{bmatrix}$$

Matrix view ...

- and error terms ε_i
- means (or intercepts) $\boldsymbol{\alpha}$

- The loading matrix is $\boldsymbol{\Lambda}$ (capital lambda), it has one column per latent variable.
  - possible assumed loadings

Completely General

$$\begin{bmatrix} \lambda_{11} & \lambda_{12} \\ \lambda_{21} & \lambda_{22} \\ \lambda_{31} & \lambda_{32} \\ \lambda_{41} & \lambda_{42} \\ \lambda_{51} & \lambda_{52} \\ \lambda_{61} & \lambda_{62} \end{bmatrix}$$

More usual has plenty of 0's:

$$\begin{bmatrix} \lambda_{11} & 0 \\ \lambda_{21} & 0 \\ \lambda_{31} & 0 \\ 0 & \lambda_{42} \\ 0 & \lambda_{52} \\ 0 & \lambda_{62} \end{bmatrix}$$

- And the whole model would be

$$\mathbf{x}_i = \boldsymbol{\alpha} + \boldsymbol{\Lambda}\boldsymbol{\xi}_i + \boldsymbol{\varepsilon}_i$$

Outline

Did you notice loadings = 1.0 in CFA output?

- It is not possible to estimate, at the same time, all of the loadings and variances.
- We must set the scale of some coefficients, then we estimate the others
- lavaan used the "marker variable" method (Lindell & Whitney, 2001), because for each latent variable a single variable was selected to determine the scale.
- Another commonly used method is to fix the variance of the latent variables at 1.0 (thus allowing the loadings to float freely)

Obtain "standardized latent variable" estimates

```
cfa3 <- cfa(cfa.mod, data = dat, std.lv = TRUE)
```

```
summary(cfa3)
```

```
lavaan 0.6-3 ended normally after 58 iterations

  Optimization method                           NLMINB
  Number of free parameters                         40

  Number of observations                           380

  Estimator                                         ML
  Model Fit Test Statistic                     106.847
  Degrees of freedom                                80
  P-value (Chi-square)                           0.024

Parameter Estimates:

  Information                                 Expected
  Information saturated (h1) model          Structured
  Standard Errors                             Standard

Latent Variables:
```

Obtain "standardized latent variable" estimates ...

```
                   Estimate  Std.Err  z-value  P(>|z|)
  Agency =~
    Agency1           0.466    0.021   22.560    0.000
    Agency2           0.492    0.021   23.774    0.000
    Agency3           0.497    0.022   22.815    0.000
  Intrinsic =~
    Intrin1           0.541    0.039   13.960    0.000
    Intrin2           0.581    0.043   13.489    0.000
    Intrin3           0.615    0.038   16.201    0.000
  Extrinsic =~
    Extrin1           0.388    0.022   17.606    0.000
    Extrin2           0.456    0.027   17.168    0.000
    Extrin3           0.471    0.026   18.100    0.000
  Positive =~
    PosAFF1           0.569    0.028   20.132    0.000
    PosAFF2           0.603    0.029   21.150    0.000
    PosAFF3           0.632    0.029   21.648    0.000
  Negative =~
    NegAFF1           0.634    0.029   21.670    0.000
    NegAFF2           0.585    0.027   21.457    0.000
    NegAFF3           0.598    0.026   22.805    0.000

Covariances:
                   Estimate  Std.Err  z-value  P(>|z|)
  Agency ~~
    Intrinsic         0.507    0.047   10.782    0.000
```

Obtain "standardized latent variable" estimates ...

```
    Extrinsic          0.270    0.054    5.028    0.000
    Positive           0.302    0.051    5.950    0.000
    Negative           0.017    0.055    0.312    0.755
  Intrinsic ~~
    Extrinsic         -0.028    0.063   -0.441    0.659
    Positive           0.414    0.052    7.918    0.000
    Negative           0.021    0.060    0.341    0.733
  Extrinsic ~~
    Positive          -0.029    0.058   -0.503    0.615
    Negative           0.209    0.056    3.764    0.000
  Positive ~~
    Negative          -0.071    0.056   -1.274    0.203

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .Agency1           0.048    0.005    9.649    0.000
   .Agency2           0.036    0.005    7.604    0.000
   .Agency3           0.051    0.005    9.278    0.000
   .Intrin1           0.298    0.029   10.204    0.000
   .Intrin2           0.386    0.036   10.620    0.000
   .Intrin3           0.214    0.029    7.474    0.000
   .Extrin1           0.080    0.009    8.924    0.000
   .Extrin2           0.123    0.013    9.465    0.000
   .Extrin3           0.103    0.012    8.264    0.000
   .PosAFF1           0.121    0.012    9.921    0.000
   .PosAFF2           0.104    0.012    8.615    0.000
```

Obtain "standardized latent variable" estimates ...

```
    .PosAFF3          0.099    0.013    7.869    0.000
    .NegAFF1          0.107    0.012    9.202    0.000
    .NegAFF2          0.096    0.010    9.498    0.000
    .NegAFF3          0.067    0.009    7.369    0.000
     Agency           1.000
     Intrinsic        1.000
     Extrinsic        1.000
     Positive         1.000
     Negative         1.000
```

Note: The variances of Agency, Intrinsic, etc, are 1.0 because we selected std.lv = TRUE

Outline

Estimation methods

The `estimator` argument in lavaan functions determines the method of estimation

- ML: Maximum Wishart likelihood of the complete cases (listwise-delete missings)
  - Choose estimates to make the observed covariance matrix most likely
  - If `missing = "fiml"`, then Full Information Maximum Likelihood is used to avoid listwise deletion
- WLSMV: weighted least squares with a mean and variance adjustment
  - Standard error estimates that are "robust" to violations of multivariate normality

Bootstrap estimates are also included

- The robust estimators of the standard errors are approximate, but they are widely used.
- lavaan functions include a bootstrap argument
- Why? In small samples, or when parameter distributions are unknown, this is a popular method to evaluate uncertainty.
- The bootstrap estimator will draw repeated random samples and re-estimate the model.

KU

summary

- The `summary()` method in lavaan generates estimate tables and some summary information.
- Observant readers might have noticed that the default output, obtained from

```
summary(cfa1)
```

  did not include the "fit measures" that are often offered with CFA and SEM output in other software.
- Two ways to deal with that.
  1. Requests fits in summary function
  2. Inspect model fit measures separately.

summary with additional details

- First, we will insert a request for **fit measures** and the **rsquare** into the summary output. The result looks quite a bit more like Mplus. We select ci=FALSE to reduce the width of the output

```
summary(cfa3, rsquare = TRUE, fit.measures =
    TRUE, ci = FALSE)
```

```
lavaan 0.6-3 ended normally after 58 iterations

  Optimization method                           NLMINB
  Number of free parameters                         40

  Number of observations                           380

  Estimator                                         ML
  Model Fit Test Statistic                     106.847
  Degrees of freedom                                80
  P-value (Chi-square)                           0.024

Model test baseline model:

  Minimum Function Test Statistic             3749.411
  Degrees of freedom                                105
```

summary with additional details ...

```
  P-value                                            0.000

User model versus baseline model:

  Comparative Fit Index (CFI)                        0.993
  Tucker-Lewis Index (TLI)                           0.990

Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)              -3699.110
  Loglikelihood unrestricted model (H1)      -3645.686

  Number of free parameters                         40
  Akaike (AIC)                                7478.219
  Bayesian (BIC)                              7635.826
  Sample-size adjusted Bayesian (BIC)         7508.914

Root Mean Square Error of Approximation:

  RMSEA                                              0.030
  90 Percent Confidence Interval          0.011  0.044
  P-value RMSEA <= 0.05                              0.994

Standardized Root Mean Square Residual:

  SRMR                                               0.031
```

summary with additional details ...

```
Parameter Estimates:

  Information                                 Expected
  Information saturated (h1) model          Structured
  Standard Errors                             Standard

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)
  Agency =~
    Agency1           0.466    0.021   22.560    0.000
    Agency2           0.492    0.021   23.774    0.000
    Agency3           0.497    0.022   22.815    0.000
  Intrinsic =~
    Intrin1           0.541    0.039   13.960    0.000
    Intrin2           0.581    0.043   13.489    0.000
    Intrin3           0.615    0.038   16.201    0.000
  Extrinsic =~
    Extrin1           0.388    0.022   17.606    0.000
    Extrin2           0.456    0.027   17.168    0.000
    Extrin3           0.471    0.026   18.100    0.000
  Positive =~
    PosAFF1           0.569    0.028   20.132    0.000
    PosAFF2           0.603    0.029   21.150    0.000
    PosAFF3           0.632    0.029   21.648    0.000
  Negative =~
```

summary with additional details ...

```
    NegAFF1           0.634    0.029   21.670    0.000
    NegAFF2           0.585    0.027   21.457    0.000
    NegAFF3           0.598    0.026   22.805    0.000

Covariances:
                   Estimate  Std.Err  z-value  P(>|z|)
  Agency ~~
    Intrinsic         0.507    0.047   10.782    0.000
    Extrinsic         0.270    0.054    5.028    0.000
    Positive          0.302    0.051    5.950    0.000
    Negative          0.017    0.055    0.312    0.755
  Intrinsic ~~
    Extrinsic        -0.028    0.063   -0.441    0.659
    Positive          0.414    0.052    7.918    0.000
    Negative          0.021    0.060    0.341    0.733
  Extrinsic ~~
    Positive         -0.029    0.058   -0.503    0.615
    Negative          0.209    0.056    3.764    0.000
  Positive ~~
    Negative         -0.071    0.056   -1.274    0.203

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .Agency1           0.048    0.005    9.649    0.000
   .Agency2           0.036    0.005    7.604    0.000
   .Agency3           0.051    0.005    9.278    0.000
```

summary with additional details ...

```
    .Intrin1          0.298    0.029   10.204    0.000
    .Intrin2          0.386    0.036   10.620    0.000
    .Intrin3          0.214    0.029    7.474    0.000
    .Extrin1          0.080    0.009    8.924    0.000
    .Extrin2          0.123    0.013    9.465    0.000
    .Extrin3          0.103    0.012    8.264    0.000
    .PosAFF1          0.121    0.012    9.921    0.000
    .PosAFF2          0.104    0.012    8.615    0.000
    .PosAFF3          0.099    0.013    7.869    0.000
    .NegAFF1          0.107    0.012    9.202    0.000
    .NegAFF2          0.096    0.010    9.498    0.000
    .NegAFF3          0.067    0.009    7.369    0.000
     Agency           1.000
     Intrinsic        1.000
     Extrinsic        1.000
     Positive         1.000
     Negative         1.000

R-Square:
                   Estimate
     Agency1          0.819
     Agency2          0.871
     Agency3          0.830
     Intrin1          0.495
     Intrin2          0.467
     Intrin3          0.639
```

summary with additional details ...

```
    Extrin1           0.653
    Extrin2           0.628
    Extrin3           0.682
    PosAFF1           0.728
125 PosAFF2           0.778
    PosAFF3           0.802
    NegAFF1           0.790
    NegAFF2           0.780
    NegAFF3           0.842
```

- What other details could we ask for? We checked the lavaan source code, where we find the summary function allows these flags:
  - header (default: TRUE)
  - fit.measures (default: FALSE)
  - estimates (default: TRUE)
  - ci (default: FALSE)
  - fmi (default: FALSE)
  - standardized (default: FALSE)
  - rsquare (default: FALSE)
  - std.nox (default: FALSE)
  - modindices (default: FALSE)

KU

parameterEstimates

- lavaan's `summary` output is created by separate functions which can be accessed directly.
- `parameterEstimates()` is doing most of the actual work.
- The relevant arguments for `parameterEstimates()` are
  - se (default: TRUE) Show standard errors?
  - zstat (default: TRUE) Show Z, ratio of estimate to standard error
  - pvalue (default: TRUE) Show p value
  - ci (default: TRUE) Show confidence interval
  - level (default: 0.95) Confidence level required to calculate ci
  - boot.ci.type: (default: "perc") Confidence interval for bootstrapped models
  - standardized (default: FALSE) Add standardized parameter estimates
  - fmi (default: FALSE) Show fraction of missing information, for "FIML" models
  - remove.system.eq (default: TRUE) Hide user-constrained parameters
  - remove.eq (default.eq: TRUE) Hide system-generated constraints
  - remove.ineq (default: TRUE) Hide inequality constraints

KU

parameterEstimates ...

- remove.def (default: FALSE) Hide parameter definitions
- rsquare (default: FALSE) Add rows for the R-square

- The default output was too wide for these slides, so we don't look at p or ci. Here are the first 15 lines:

```
parameterEstimates(cfa3, pvalue = FALSE, ci =
    FALSE)
```

```
          lhs op      rhs   est    se      z
1      Agency =~ Agency1 0.466 0.021 22.560
2      Agency =~ Agency2 0.492 0.021 23.774
3      Agency =~ Agency3 0.497 0.022 22.815
4   Intrinsic =~ Intrin1 0.541 0.039 13.960
5   Intrinsic =~ Intrin2 0.581 0.043 13.489
6   Intrinsic =~ Intrin3 0.615 0.038 16.201
7   Extrinsic =~ Extrin1 0.388 0.022 17.606
8   Extrinsic =~ Extrin2 0.456 0.027 17.168
9   Extrinsic =~ Extrin3 0.471 0.026 18.100
10   Positive =~ PosAFF1 0.569 0.028 20.132
11   Positive =~ PosAFF2 0.603 0.029 21.150
12   Positive =~ PosAFF3 0.632 0.029 21.648
13   Negative =~ NegAFF1 0.634 0.029 21.670
```

parameterEstimates ...

```
15  14   Negative =~ NegAFF2 0.585 0.027 21.457
    15   Negative =~ NegAFF3 0.598 0.026 22.805
```

parTable

- `parTable()` : create a data.frame object that holds the estimated values, with one row per parameter:

```
cfa3.df <- parTable(cfa3)
head(cfa3.df, 10)
```

```
   id        lhs op     rhs user block group free ustart exo label
       plabel start   est    se
1   1     Agency =~ Agency1    1     1     1    1     NA   0
    .p1. 0.466 0.466 0.021
2   2     Agency =~ Agency2    1     1     1    2     NA   0
    .p2. 0.492 0.492 0.021
3   3     Agency =~ Agency3    1     1     1    3     NA   0
    .p3. 0.497 0.497 0.022
4   4 Intrinsic =~ Intrin1    1     1     1    4     NA   0
    .p4. 0.538 0.541 0.039
5   5 Intrinsic =~ Intrin2    1     1     1    5     NA   0
    .p5. 0.591 0.581 0.043
6   6 Intrinsic =~ Intrin3    1     1     1    6     NA   0
    .p6. 0.610 0.615 0.038
7   7 Extrinsic =~ Extrin1    1     1     1    7     NA   0
    .p7. 0.389 0.388 0.022
8   8 Extrinsic =~ Extrin2    1     1     1    8     NA   0
    .p8. 0.452 0.456 0.027
9   9 Extrinsic =~ Extrin3    1     1     1    9     NA   0
    .p9. 0.473 0.471 0.026
```

fitMeasures

- A comprehensive list of fit indicators is returned by `fitMeasures(cfa1)`

```
fitMeasures(cfa1)
```

```
               npar                fmin                chisq
                                     df              pvalue
             40.000               0.141             106.847
                                 80.000               0.024
     baseline.chisq         baseline.df     baseline.pvalue
                                    cfi                 tli
           3749.411             105.000               0.000
                                  0.993               0.990
               nnfi                rfi                 nfi
                                   pnfi                ifi
              0.990               0.963               0.972
                                  0.740               0.993
                rni                logl  unrestricted.logl
                                    aic                 bic
              0.993           -3699.110           -3645.686
                           7478.219            7635.826
             ntotal               bic2               rmsea
                    rmsea.ci.lower      rmsea.ci.upper
            380.000           7508.914               0.030
                                0.011               0.044
```

fitMeasures ...

```
       rmsea.pvalue                    rmr             rmr_nomean
                              srmr          srmr_bentler
              0.994                  0.014                  0.014
                                     0.031                  0.031
srmr_bentler_nomean                    crmr          crmr_nomean
     srmr_mplus    srmr_mplus_nomean
              0.031                  0.033                  0.033
                                     0.031                  0.031
              cn_05                  cn_01                    gfi
                                    agfi                   pgfi
            363.332                400.494                  0.964
                                   0.946                  0.643
                mfi                    ecvi
              0.965                  0.492
```

- Just a few measures? How about the CFI and RMSEA only?

```
fitMeasures(cfa1, fit.measures = c("cfi",
    "rmsea"))
```

```
  cfi rmsea
0.993 0.030
```

fitMeasures ...

- `fitMeasures`, you will see the list of all possible fit.measures is not fully documented, but it *at least* includes:

```
"cfi","tli","nnfi","pnfi","rfi","nfi",
"ifi","rmsea","rmsea.ci.lower","rmsea.ci.upper",
"rmsea.pvalue","rmr", "srmr", "wrmr","agfi", "pgfi", "mfi", "ecvi",
"baseline.chisq", "baseline.pvalue", "baseline.df"
```

CFA Commentary

- The model appears to fit well,
  - all factor loadings are significant, and the
  - standardized factor loadings indicate strong correlations between indicators and constructs.
- However, we did not explicitly model the relationships among the latent variables.
  - We don't have "Agency" as predictor of "Positive" affect.
  - The CFA fits "unstructured covariances" between latent variables, not "directional regression relationships".

KU

Other follow-up functions

- Other standard R accessor functions are available in lavaan

```
coef(sem1)
fitted(sem1)
resid(sem1)
anova(sem1)
```

- If you run those things, you will notice some wrinkles.
  - Notice that predicted observations (and residuals) are not 1-per-person
  - `anova()` returns a chi-squared test that indicates there are 0 degrees of freedom (that's an SEM concept).

A nice output table

- `semTable` has been in `kutils` package, but now it will be in `semTable` package (now on CRAN).
- Style of usage is similar to `rockchalk` package `outreg` function
- `semTable` package has a long vignette demonstrating many possibilities

```
library(kutils)
lav70 <- semTable(cfa3, fits = c("cfi", "rmsea"),
   longtable = TRUE,
          file = file.path(tmpout, "lav70"),
             print.results=FALSE)
```

```
cat(lav70)
```

	Model			
	Estimate	Std. Err.	z	p

A nice output table ...

		Factor Loadings			
Agency					
	Agency1	0.47	0.02	22.56	0.000
	Agency2	0.49	0.02	23.77	0.000
	Agency3	0.50	0.02	22.81	0.000
Intrinsic					
	Intrin1	0.54	0.04	13.96	0.000
	Intrin2	0.58	0.04	13.49	0.000
	Intrin3	0.62	0.04	16.20	0.000
Extrinsic					
	Extrin1	0.39	0.02	17.61	0.000
	Extrin2	0.46	0.03	17.17	0.000
	Extrin3	0.47	0.03	18.10	0.000
Positive					
	PosAFF1	0.57	0.03	20.13	0.000
	PosAFF2	0.60	0.03	21.15	0.000

A nice output table ...

	PosAFF3	0.63	0.03	21.65	0.000
Negative					
	NegAFF1	0.63	0.03	21.67	0.000
	NegAFF2	0.59	0.03	21.46	0.000
	NegAFF3	0.60	0.03	22.81	0.000
			Residual Variances		
	Agency1	0.05	0.00	9.65	0.000
	Agency2	0.04	0.00	7.60	0.000
	Agency3	0.05	0.01	9.28	0.000
	Intrin1	0.30	0.03	10.20	0.000
	Intrin2	0.39	0.04	10.62	0.000
	Intrin3	0.21	0.03	7.47	0.000
	Extrin1	0.08	0.01	8.92	0.000
	Extrin2	0.12	0.01	9.46	0.000
	Extrin3	0.10	0.01	8.26	0.000
	PosAFF1	0.12	0.01	9.92	0.000
	PosAFF2	0.10	0.01	8.62	0.000

A nice output table ...

PosAFF3	0.10	0.01	7.87	0.000
NegAFF1	0.11	0.01	9.20	0.000
NegAFF2	0.10	0.01	9.50	0.000
NegAFF3	0.07	0.01	7.37	0.000
		Latent Variances		
Agency	1.00^{+}			
Intrinsic	1.00^{+}			
Extrinsic	1.00^{+}			
Positive	1.00^{+}			
Negative	1.00^{+}			
		Latent Covariances		
Agency w/Intrinsic	0.51	0.05	10.78	0.000
Agency w/Extrinsic	0.27	0.05	5.03	0.000
Agency w/Positive	0.30	0.05	5.95	0.000
Agency w/Negative	0.02	0.05	0.31	0.755
Intrinsic w/Extrinsic	−0.03	0.06	−0.44	0.659
Intrinsic w/Positive	0.41	0.05	7.92	0.000

KU

A nice output table ...

Intrinsic w/Negative	0.02	0.06	0.34	0.733
Extrinsic w/Positive	−0.03	0.06	−0.50	0.615
Extrinsic w/Negative	0.21	0.06	3.76	0.000
Positive w/Negative	−0.07	0.06	−1.27	0.203
		Fit Indices		
CFI	0.99			
RMSEA	0.03			

[+] Fixed parameter

Remove unwanted "Latent Variances" component

- See documentation for `?semTable`, it lists the parameter sets that can be included.
- I don't need to see both SE and T, and never want p values.

```
lav72 <- semTable(cfa3, paramSets = c("loadings",
    "intercepts", "residualvariances",
    "latentcovariances", "fits"),  columns =
    c("estsestars"), fits = c("cfi", "rmsea"),
    longtable = TRUE, file = file.path(tmpout,
    "lav72"), print.results=FALSE)
```

/framebreak

```
cat(lav72)
```

Model

Remove unwanted "Latent Variances" component ...

		Estimate(Std.Err.)
		Factor Loadings
Agency		
	Agency1	0.47(0.02)***
	Agency2	0.49(0.02)***
	Agency3	0.50(0.02)***
Intrinsic		
	Intrin1	0.54(0.04)***
	Intrin2	0.58(0.04)***
	Intrin3	0.62(0.04)***
Extrinsic		
	Extrin1	0.39(0.02)***
	Extrin2	0.46(0.03)***
	Extrin3	0.47(0.03)***
Positive		
	PosAFF1	0.57(0.03)***

Remove unwanted "Latent Variances" component ...

	PosAFF2	0.60(0.03)***
	PosAFF3	0.63(0.03)***
Negative		
	NegAFF1	0.63(0.03)***
	NegAFF2	0.59(0.03)***
	NegAFF3	0.60(0.03)***
	Residual Variances	
	Agency1	0.05(0.00)***
	Agency2	0.04(0.00)***
	Agency3	0.05(0.01)***
	Intrin1	0.30(0.03)***
	Intrin2	0.39(0.04)***
	Intrin3	0.21(0.03)***
	Extrin1	0.08(0.01)***
	Extrin2	0.12(0.01)***
	Extrin3	0.10(0.01)***
	PosAFF1	0.12(0.01)***

Remove unwanted "Latent Variances" component ...

PosAFF2	0.10(0.01)***
PosAFF3	0.10(0.01)***
NegAFF1	0.11(0.01)***
NegAFF2	0.10(0.01)***
NegAFF3	0.07(0.01)***
	Latent Covariances
Agency w/Intrinsic	0.51(0.05)***
Agency w/Extrinsic	0.27(0.05)***
Agency w/Positive	0.30(0.05)***
Agency w/Negative	0.02(0.05)
Intrinsic w/Extrinsic	−0.03(0.06)
Intrinsic w/Positive	0.41(0.05)***
Intrinsic w/Negative	0.02(0.06)
Extrinsic w/Positive	−0.03(0.06)
Extrinsic w/Negative	0.21(0.06)***
Positive w/Negative	−0.07(0.06)
	Fit Indices

Remove unwanted "Latent Variances" component ...

CFI	0.99
RMSEA	0.03

+Fixed parameter

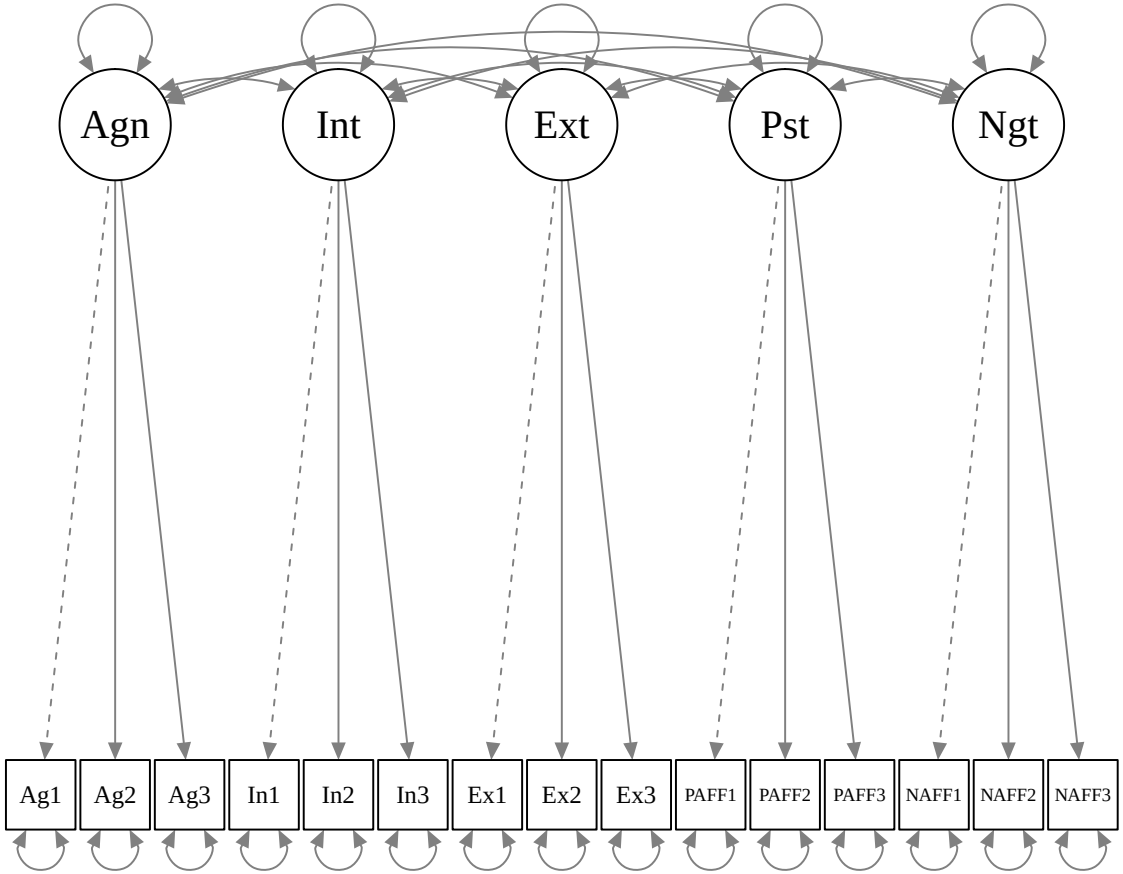
* p<0.05, ** p<0.01, *** p<0.001

Plots

- There is no plot method for a lavaan object.
- But there are other packages devoted to creating graphics for models that involve path diagrams (e.g., SEM and neural networks).
- The path diagram is visualized here with the `semPaths()` function in semPlot.

```
library(semPlot)
semPaths(cfa1, layout = "tree2")
```

Plots ...


Agn
Int
Ext
Pst
Ngt
Ag1
Ag2
Ag3
In1
In2
In3
Ex1
Ex2
Ex3
PAFF1
PAFF2
PAFF3
NAFF1
NAFF2
NAFF3

Tikz as an alternative

- Opinion: `semPlot` output will almost never be suitable for publication or professional presentation.
- We learned how to use Tikz (a LaTeX package) to make drawings that are acceptable (same as used by Ed Merkle and Yves Rosseel in their work).
- Our Tikz example collection has many SEM illustrations
- `https://gitlab.crmda.ku.edu/crmda/tikz` is a Git repo that you can
  - gaze upon with admiration and joy (scroll down)
  - download,

Tikz: what is the big idea

- LaTeX "standalone" document, only for purpose of creating PDF of drawing
- Simple style to represent boxes, label, arrows, etc.
- The repository folders have full sized pdf output and all you need to reproduce those files.

What is this section about?

- We have a more detailed example of multi-group CFA in the semexamples archive
- This section is mostly about using the following lavaan features
  1. Named groups
  2. Named group parameters
  3. `anova()` function to conduct likelihood ratio (LR) test

Moderator effects

- Moderators are categorical predictors.
- In a regression context, suppose $Agency_i$ is a continuous predictor of $Positive_i$.

$$Positive_i = \beta_0 + \beta_1 Agency_i, \epsilon_i \sim N(0, \sigma^2_\epsilon) \quad (1)$$

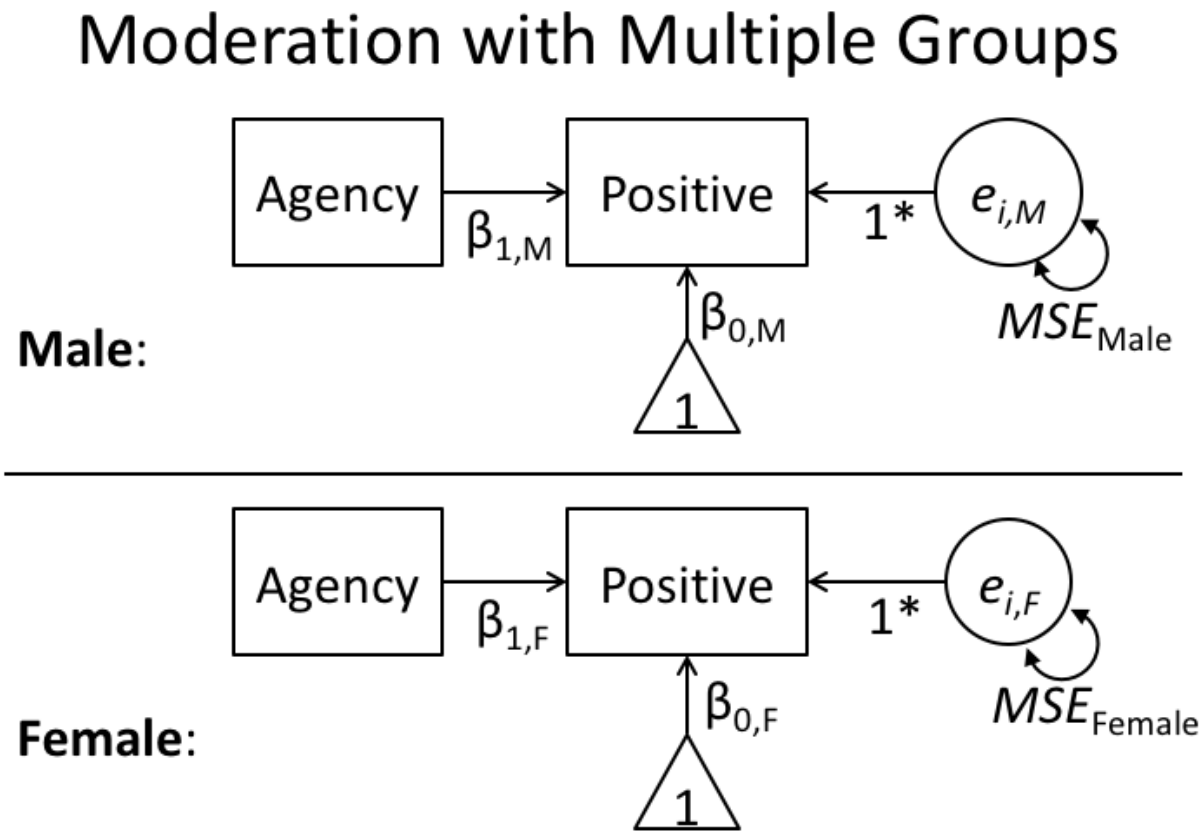
- But we wonder if a categorical variable, $female_i$, causes a change in both the intercept and the slope:

$$Positive_i = \beta_0 + \beta_1 Agency_i + \beta_2 Female_i + \beta_3 Agency_i \cdot Female_i, \epsilon_i \sim N(0, \sigma^2_\epsilon) \quad (2)$$

- Here, the gender is a "**moderator**" of the agency effect

SEM view of the moderator variable

- This sketch segregates the 2 genders entirely; we are estimating 2 separate sets of coefficients.



2 fits

- Moderation analysis focuses on the differences between groups. In this case, males and females.
- When one of the predictors in an interaction is categorical (e.g., gender), the sem "measurement invariance" approach would lead us to compare
  - a model in which the coefficients for the two groups are assumed to be entirely different, against
  - a simpler model in which some coefficients might be the same
  - here, we "stay on the surface" just to show lavaan and the hypo-test.

lavaan syntax

- The group argument in lavaan requests different loadings and intercepts for each group.
- We won't run the basic model

```
moderate.mod1 <- 'PosAFF1 ~ Agency1'
out.mod1 <- sem(moderate.mod1, data = dat,
    group = "Sex")
```

- We want to name the coefficients "c(a1, a2)" so that we have a handle for them in later exercises.

```
moderate.mod1 <- 'PosAFF1 ~ c(a1, a2)*Agency1'
out.mod1 <- sem(moderate.mod1, data = dat, group
    = "Sex")
summary(out.mod1)
```

lavaan syntax ...

```
lavaan 0.6-3 ended normally after 20 iterations

  Optimization method                           NLMINB
  Number of free parameters                          6

  Number of observations per group
  1                                                195
  2                                                185

  Estimator                                         ML
  Model Fit Test Statistic                       0.000
  Degrees of freedom                                 0
  Minimum Function Value                0.0000000000000

Chi-square for each group:

  1                                              0.000
  2                                              0.000

Parameter Estimates:

  Information                                 Expected
  Information saturated (h1) model          Structured
  Standard Errors                             Standard
```

lavaan syntax ...

```
Group 1 [1]:

Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)
  PosAFF1 ~
    Agency1   (a1)    0.326    0.085    3.862    0.000

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)
   .PosAFF1           2.317    0.212   10.909    0.000

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .PosAFF1           0.370    0.037    9.874    0.000


Group 2 [2]:

Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)
  PosAFF1 ~
    Agency1   (a2)    0.214    0.099    2.161    0.031

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)
```

lavaan syntax ...

```
   .PosAFF1          2.634    0.245   10.737    0.000

Variances:
                  Estimate  Std.Err  z-value  P(>|z|)
   .PosAFF1          0.481    0.050    9.618    0.000
```

lavaan syntax

- This model constrains the loadings for agency to be equal to the same value for both sexes

```
moderate.mod2 <- 'PosAFF1 ~ c(a1, a1)*Agency1'
out.mod2 <- sem(moderate.mod2, data = dat, group
    = "Sex")
summary(out.mod2)
```

```
lavaan 0.6-3 ended normally after 17 iterations

  Optimization method                           NLMINB
  Number of free parameters                          6
  Number of equality constraints                     1

  Number of observations per group
  1                                                195
  2                                                185

  Estimator                                         ML
  Model Fit Test Statistic                       0.747
  Degrees of freedom                                 1
  P-value (Chi-square)                           0.388
```

lavaan syntax ...

```
Chi-square for each group:

  1                                                     0.315
  2                                                     0.432

Parameter Estimates:

  Information                                         Expected
  Information saturated (h1) model                  Structured
  Standard Errors                                     Standard


Group 1 [1]:

Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)
  PosAFF1 ~
    Agency1   (a1)    0.279    0.064    4.336    0.000

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)
   .PosAFF1           2.434    0.164   14.829    0.000

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
```

lavaan syntax ...

```
   .PosAFF1          0.370    0.038    9.874    0.000


Group 2 [2]:

Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)
  PosAFF1 ~
    Agency1    (a1)    0.279    0.064    4.336    0.000

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)
   .PosAFF1           2.476    0.164   15.087    0.000

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .PosAFF1           0.482    0.050    9.618    0.000
```

Obtain a likelihood ratio test

- The `anova()` function is a generic in R, it is used in many contexts for comparing models (even models that have nothing to do with ANOVA).
- In SEM, it is used to compare models, to conduct an assessment of the extent to which a simpler model fits the data as well as a more detailed model.
- Lavaan's implementation of anova() leads to a likelihood ratio test to compare the 2 fitted models

```
anova(out.mod1, out.mod2)
```

```
Chi Square Difference Test

          Df     AIC     BIC  Chisq Chisq diff Df diff Pr(>Chisq)
out.mod1   0 761.03 784.67 0.0000
out.mod2   1 759.77 779.47 0.7467    0.74668       1     0.3875
```

We have much more work to do

- Long term goal: Understanding (regression) relationships among latent variables
- This is a modern approach to the traditional "path analysis" that has been popular in sociology and psychology for many years.
- The new implementation, dubbed LISREL in the 1970s, has grown and transformed itself.

KU

Structural Equation Modeling

- In Psychology, SEM has been an area of tremendous growth since 1980.
- The "gold standard" software for SEM modeling is Mplus, although lavaan has succeeded in "matching" side-by-side many of the calculations.

References

R Core Team (2017). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Session

```
sessionInfo()
```

```
R version 3.6.0 (2019-04-26)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 19.04

Matrix products: default
BLAS:   /usr/lib/x86_64-linux-gnu/atlas/libblas.so.3.10.3
LAPACK: /usr/lib/x86_64-linux-gnu/atlas/liblapack.so.3.10.3

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
     LC_TIME=en_US.UTF-8
 [4] LC_COLLATE=en_US.UTF-8     LC_MONETARY=en_US.UTF-8
     LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C                  LC_ADDRESS=C
[10] LC_TELEPHONE=C             LC_MEASUREMENT=en_US.UTF-8
    LC_IDENTIFICATION=C

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] semPlot_1.1.1 kutils_1.69   lavaan_0.6-3
```

Session ...

```
loaded via a namespace (and not attached):
 [1] nlme_3.1-140         RColorBrewer_1.1-2   mi_1.0
     tools_3.6.0
 [5] backports_1.1.4      R6_2.4.0             rpart_4.1-15
     d3Network_0.5.2.1
 [9] Hmisc_4.2-0          lazyeval_0.2.2       colorspace_1.4-1
     nnet_7.3-12
[13] tidyselect_0.2.5     gridExtra_2.3        mnormt_1.5-5
     compiler_3.6.0
[17] qgraph_1.6.2         fdrtool_1.2.15       htmlTable_1.13.1
     regsem_1.3.9
[21] scales_1.0.0         checkmate_1.9.3      psych_1.8.12
     pbapply_1.4-0
[25] sem_3.1-9            stringr_1.4.0        digest_0.6.18
     pbivnorm_0.6.0
[29] foreign_0.8-71       minqa_1.2.4          base64enc_0.1-3
     jpeg_0.1-8
[33] pkgconfig_2.0.2      htmltools_0.3.6      lme4_1.1-21
     lisrelToR_0.1.4
[37] htmlwidgets_1.3      rlang_0.3.4          rstudioapi_0.10
     huge_1.3.2
[41] gtools_3.8.1         acepack_1.4.1        dplyr_0.8.1
     zip_2.0.2
[45] magrittr_1.5         OpenMx_2.13.2        Formula_1.2-3
     Matrix_1.2-17
```

Session ...

```
[49] Rcpp_1.0.1             munsell_0.5.0        abind_1.4-5
     rockchalk_1.8.144
[53] stringi_1.4.3          whisker_0.3-2        carData_3.0-2
     MASS_7.3-51.4
[57] plyr_1.8.4             matrixcalc_1.0-3     grid_3.6.0
     parallel_3.6.0
[61] crayon_1.3.4           lattice_0.20-38      splines_3.6.0
     knitr_1.22
[65] pillar_1.4.0           igraph_1.2.4.1       boot_1.3-22
     rjson_0.2.20
[69] corpcor_1.6.9          BDgraph_2.59         reshape2_1.4.3
     stats4_3.6.0
[73] XML_3.98-1.19          glue_1.3.1           latticeExtra_0.6-28
     data.table_1.12.2
[77] png_0.1-7              nloptr_1.2.1         gtable_0.3.0
     purrr_0.3.2
[81] assertthat_0.2.1       ggplot2_3.1.1        xfun_0.7
     openxlsx_4.1.0
[85] xtable_1.8-4           semTools_0.5-1       coda_0.19-2
     Rsolnp_1.16
[89] survival_2.44-1.1      glasso_1.10          truncnorm_1.0-8
     tibble_2.1.1
[93] arm_1.10-1             ggm_2.3              cluster_2.0.9
```

Session ...

```
## Don't delete this. It puts the interactive
    session options
## back the way they were. If this is compiled
    within a session
## it is vital to do this.
options(opts.orig)
> options(par.orig)
```