Structural Equation Modeling and Path Analysis

NRES 746

Fall 2018

Presenters: Maya Zawlodzki, Stephanie Coronado, Stephanie Cole, Elaine Chu, Chris Wolfe

For those wishing to follow along with the R-based demo in class, click here for the companion R script for this lecture.

For those wishing to follow along with the slide presentation, click here for the Google Sheets component of this lecture.

Set-Up

There are a LOT of packages / libraries used in this tutorial. So load them sooner than later!

```
## Install packages
#install.packages("OpenMx")
#install.packages("lavaan")
#install.packages("semPlot")
#install.packages("tidyverse")
#install.packages("GGally")

## Load libraries
suppressMessages(library(lavaan))
suppressMessages(library(OpenMx))
suppressMessages(library(semPlot))
suppressMessages(library(tidyverse))
suppressMessages(library(GGally))
```

Path Analysis

- This tutorial was adapted from "Introduction to Path Analysis in R" by Thomas Bihansky (2017)

Modeling

Because SEM is method of model verification, we need to have an initial model in mind. Keep in mind that SEM is inherently *causal*. Therefore, our results will depend on our research design.

The first step is to look at our data.

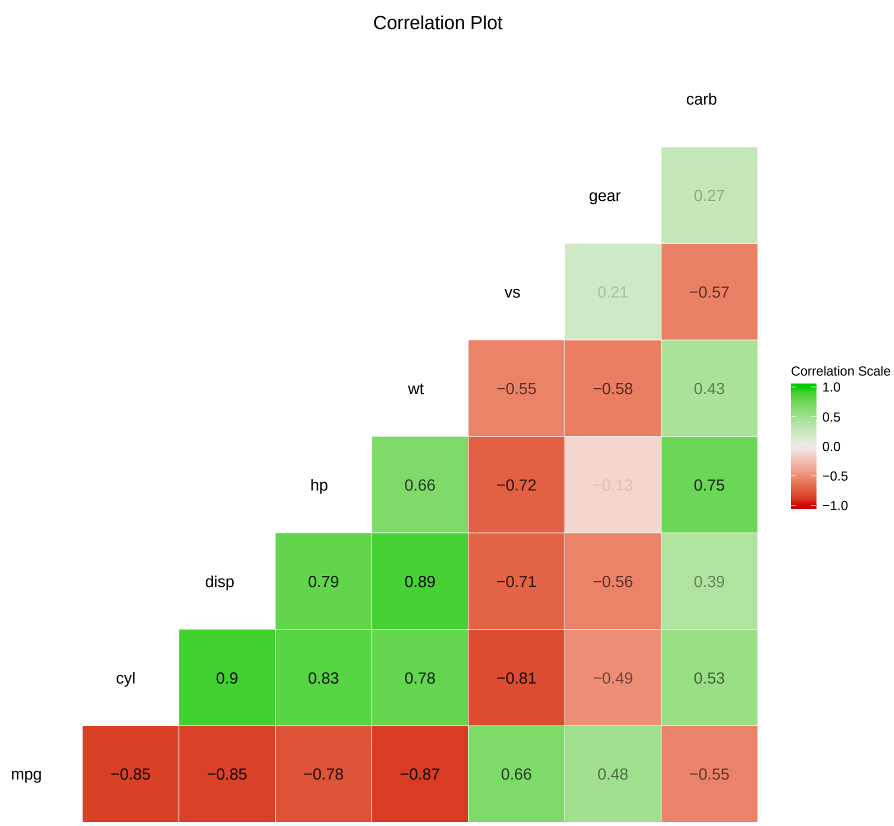
```
## View your data
summary(mtcars)
```

```
##       mpg             cyl             disp             hp       
##  Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0  
##  1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5  
##  Median :19.20   Median :6.000   Median :196.3   Median :123.0  
##  Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7  
##  3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0  
```

```
##  Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0
##       drat             wt              qsec             vs
##  Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000
##  1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000
##  Median :3.695   Median :3.325   Median :17.71   Median :0.0000
##  Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375
##  3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000
##  Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.   :1.0000
##        am              gear            carb
##  Min.   :0.0000   Min.   :3.000   Min.   :1.000
##  1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000
##  Median :0.0000   Median :4.000   Median :2.000
##  Mean   :0.4062   Mean   :3.688   Mean   :2.812
##  3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000
##  Max.   :1.0000   Max.   :5.000   Max.   :8.000
```

We can use a correlation plot to explore the relationships between the variables in our data:

```
ggcorr(mtcars[-c(5,7,9)], nbreaks=NULL, label=T, low="red3", high="green3", label_round=2, name="Correla
     ggtitle(label="Correlation Plot") +
     theme(plot.title=element_text(hjust=0.6))
```



From our summary and correlation plot, we can decide what variables we want to use in our model:

dependent variable:

- mpg

independent variables:

- hp
- cyl
- disp
- carb
- am
- wt

First, we could evaluate this relationship using a simple linear regression:

```
## Simple Linear Model:
linear <- lm(mpg ~ hp+ cyl + disp + carb + am + wt, data=mtcars)

linear
```

```
##
## Call:
## lm(formula = mpg ~ hp + cyl + disp + carb + am + wt, data = mtcars)
##
## Coefficients:
## (Intercept)           hp          cyl         disp         carb
##   36.988646    -0.020796    -0.996962     0.006998    -0.320956
##          am           wt
##    1.904625    -2.848489
```

With our generalized linear model, we have the best coefficients (beta-values) for our relationship.

But what if the relationships between our independent variables are more complex?

We can explore these complexites by applying a structure to our model. To do so, we are **applying causality**.

For that, we could draw a model:

Now, lets code our model:

```
## Set up your model in R
model = '
     # Blue Relationship
     mpg ~ hp + cyl + disp + carb + am + wt

     # Green Relationship
     hp ~ cyl + disp + carb
'
```

Test Your Model

After we have our model coded in our global environment, we can test our model to see how well it works using the sem() function:

```
## run sem()
path <- sem(model, data=mtcars)
```

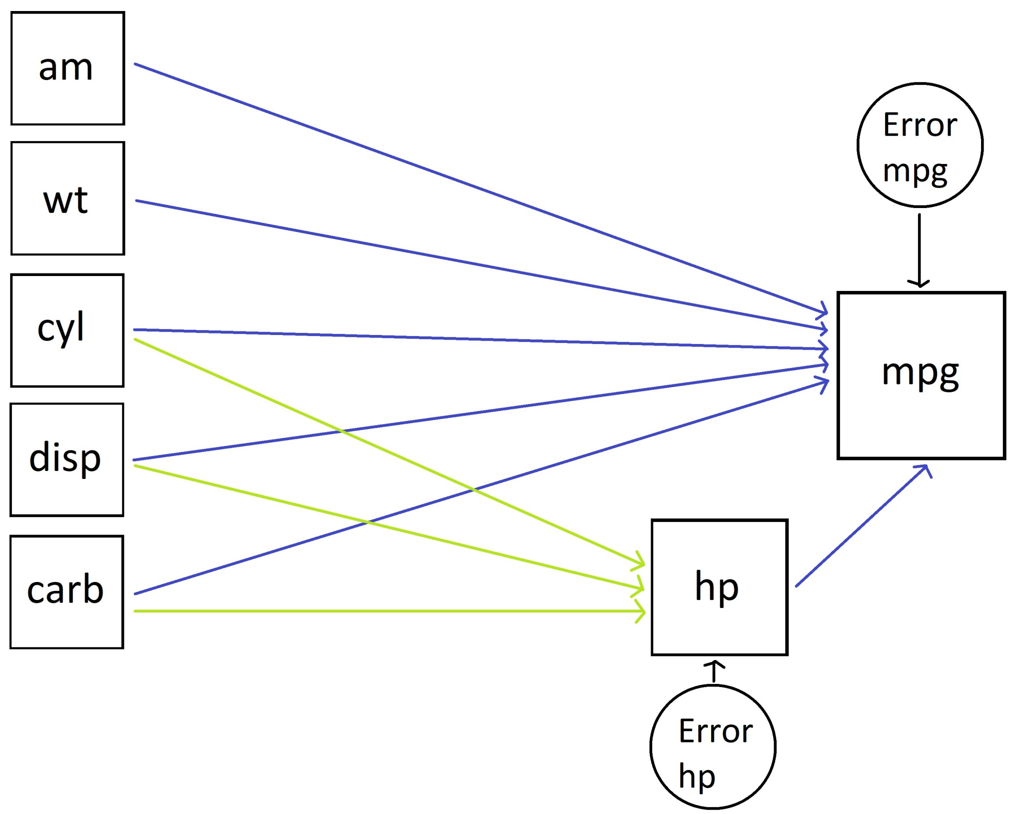


Figure 1:

```r
path
```

```
## lavaan (0.6-1) converged normally after  59 iterations
##
##   Number of observations                            32
##
##   Estimator                                         ML
##   Model Fit Test Statistic                       7.575
##   Degrees of freedom                                 2
##   P-value (Chi-square)                           0.023
```

This simple output from returning "path" provides us with a quick evaluation of our model. Here, we can do a basic evaluation of how well our model has worked, by looking at our "Model Fit Test Statistic", which is a Chi-Square statistic. Our "Degrees of Freedom" tells us how many potential paths were *NOT* used in our model. Additionally, our p-value lets us know if the Chi-Square is statisically significant. This analysis informs us how "good" our model is. Because we reject the null hypothesis, this means that our model essentially could be better.

Interpreting Results

To get a more in-depth look at our path results, including our coefficients, we can use the summary() function:

```r
summary(path, standardized = T, fit.measures=T, rsquare=T)
```

```
## lavaan (0.6-1) converged normally after  59 iterations
##
##   Number of observations                            32
##
##   Estimator                                         ML
##   Model Fit Test Statistic                       7.575
##   Degrees of freedom                                 2
##   P-value (Chi-square)                           0.023
##
## Model test baseline model:
##
##   Minimum Function Test Statistic              132.287
##   Degrees of freedom                                11
##   P-value                                        0.000
##
## User model versus baseline model:
##
##   Comparative Fit Index (CFI)                    0.954
##   Tucker-Lewis Index (TLI)                       0.747
##
## Loglikelihood and Information Criteria:
##
##   Loglikelihood user model (H0)               -220.208
##   Loglikelihood unrestricted model (H1)       -216.420
##
##   Number of free parameters                         11
##   Akaike (AIC)                                 462.416
##   Bayesian (BIC)                               478.539
##   Sample-size adjusted Bayesian (BIC)          444.247
##
```

```
## Root Mean Square Error of Approximation:
##
##   RMSEA                                          0.295
##   90 Percent Confidence Interval          0.094  0.531
##   P-value RMSEA <= 0.05                          0.030
##
## Standardized Root Mean Square Residual:
##
##   SRMR                                           0.024
##
## Parameter Estimates:
##
##   Information                                 Expected
##   Information saturated (h1) model          Structured
##   Standard Errors                             Standard
##
## Regressions:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   mpg ~
##     hp               -0.021    0.016   -1.330    0.184   -0.021   -0.234
##     cyl              -0.997    0.642   -1.554    0.120   -0.997   -0.292
##     disp              0.007    0.012    0.585    0.559    0.007    0.142
##     carb             -0.321    0.506   -0.634    0.526   -0.321   -0.085
##     am                1.905    1.425    1.336    0.181    1.905    0.156
##     wt               -2.848    1.198   -2.378    0.017   -2.848   -0.457
##   hp ~
##     cyl               7.717    6.554    1.177    0.239    7.717    0.201
##     disp              0.233    0.087    2.666    0.008    0.233    0.421
##     carb             20.273    3.405    5.954    0.000   20.273    0.478
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .mpg               5.045    1.261    4.000    0.000    5.045    0.140
##    .hp              644.737  161.184    4.000    0.000  644.737    0.142
##
## R-Square:
##                    Estimate
##     mpg               0.860
##     hp                0.858
```

Summary Break-down

The summary can provide us with additional information about our model:

- CFI and TFI are numerical indicators of fit. A value of 0.9+ is the industry standard for a "good" model fit. The closer to 1, the better

- AIC and BIC: allows us to compare models

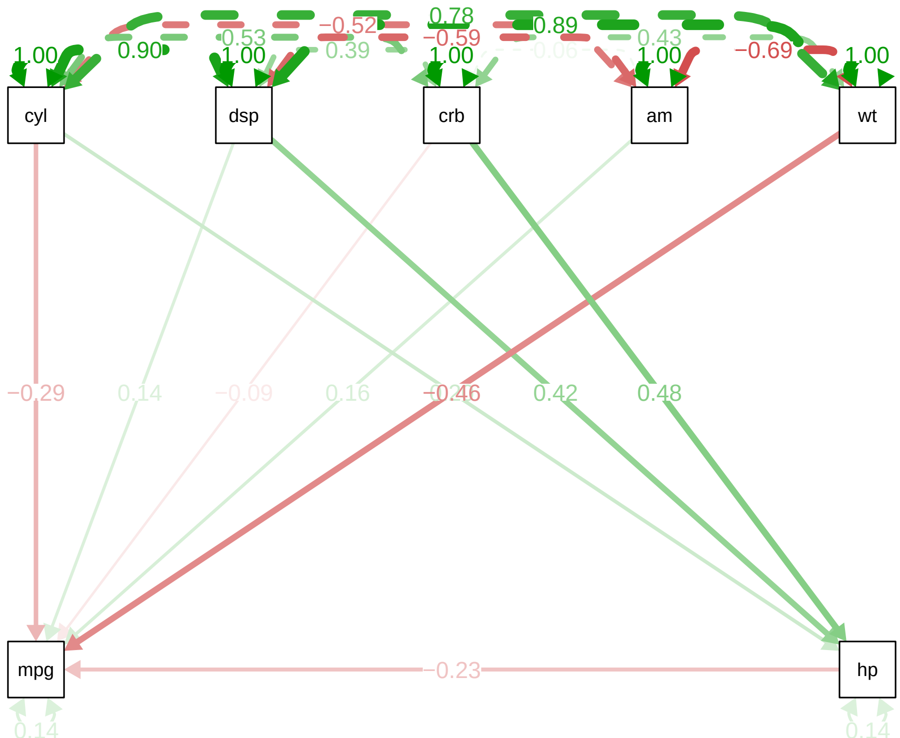
- Unstandardized coefficients: provides us with model-specific path coefficients that are of **original scale** of the data, and allows for estimation of expected values

- Standardized: Allows for comparison of path coefficients between models

- R-Square: descriptive statistic to tell us about the fit of our coefficients

Visualize Your Model!

```
## Generated path
semPaths(path, "std", layout='tree', edge.label.cex=.9, curvePivot=T)
```



The model allows us to visualize how well our model fits the data we have.

SEM with Latent Variables

- The following tutorial was adapted from Yves Rossel Lavaan Tutorial (2012)

To this point, we have demonstrated the simplest form of structural equation modeling in which only single known indicators (exogenous/independent variables) are employed for the casual model. As we have mentioned earlier, path analysis is just a special form of multiple regression and SEM.

What if some of the observed variables are related to one another in some way? This may lead to issues associated with dimensionality and multicollinearity. Therefore, a hallmark of SEM is the ability to define latent variables to account for such issues and in turn, to see how the *unobserved* (unquantified) may be related to the endogenous/response variable.

Here is the syntax for such construction in 'lavaan':

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

Where **y** is the observed response, **x** is the observed independent, and **f** is a latent variable. Notice that this is still a series of regression equations.

Any of the latent variables found in the regression formulas must be "defined" by listing their (manifestor latent) indicators. We do this by using the special operator "=~", which can be read as "is measured by."

For example, to define the three latent variables **f1**, **f2**, and **f3**, we can use the following model:

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

Additonally, SEM allows us to define variances and covariances between variables up front using the special operator "~~" or intercepts using "~". We will leave this out of the model. Let's apply our model using the lavaan package to a built-in dataset called *HolzingerSwineford1939*:

```r
#?HolzingerSwineford1939  # some information about the data

head(HolzingerSwineford1939)  # let's look at the first few lines
```

This is a classic SEM dataset that consists of mental ability test scores of seventh- and eight-grade children from two different schools.

To help with understanding the data, the descriptions of certain variables are as follows:

```
x1: visual perception
x2: cubes
x3: lozenges
x4: paragraph comprehension
x5: sentence completion
x6: word meaning
x7: speeded addition
x8: speeded counting of dots
x9: speeded discrimination straight and curved capitals
```

An SEM model that is often proposed for these 9 variables consists of three latent variables (f#), each with three indicators(y#):

1. visual factor (f1) measured by 3 variables: x1, x2, and x3

2. textual factor (f2) measured by 3 variables: x4, x5, and x6

3. speed factor (f3) measured by 3 variables: x7, x8, and x9

What if we want to observe the relationship of age, sex, school, and grade in addition to the three latent variables?

Test Your Model

```r
## Define your model:
ability_model <- '
    # regression equations
    visual ~ sex + ageyr + school + grade
    textual ~ sex + ageyr + school + grade
    speed ~ sex + ageyr + school + grade

    # latent variables
    visual =~ x1 + x2 + x3
```

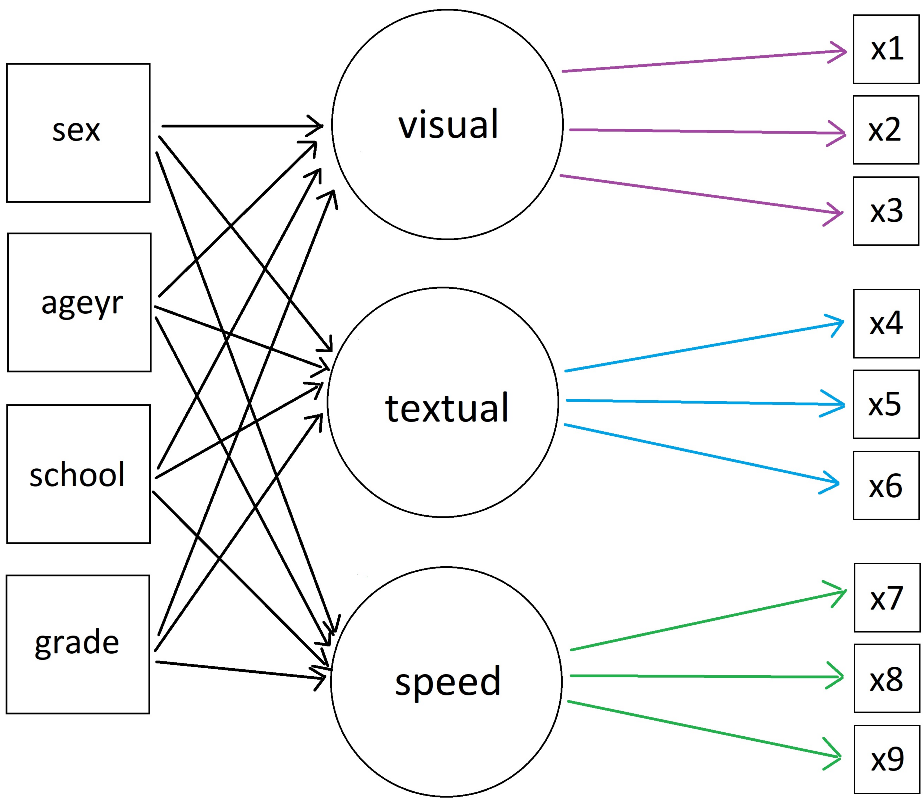

sex
ageyr
school
grade
visual
textual
speed
x1
x2
x3
x4
x5
x6
x7
x8
x9

Figure 2:

```
    textual =~ x4 + x5 + x6
    speed =~ x7 + x8 + x9

    # covariances
    visual ~~ textual
    visual ~~ speed
    textual ~~ speed
'
```

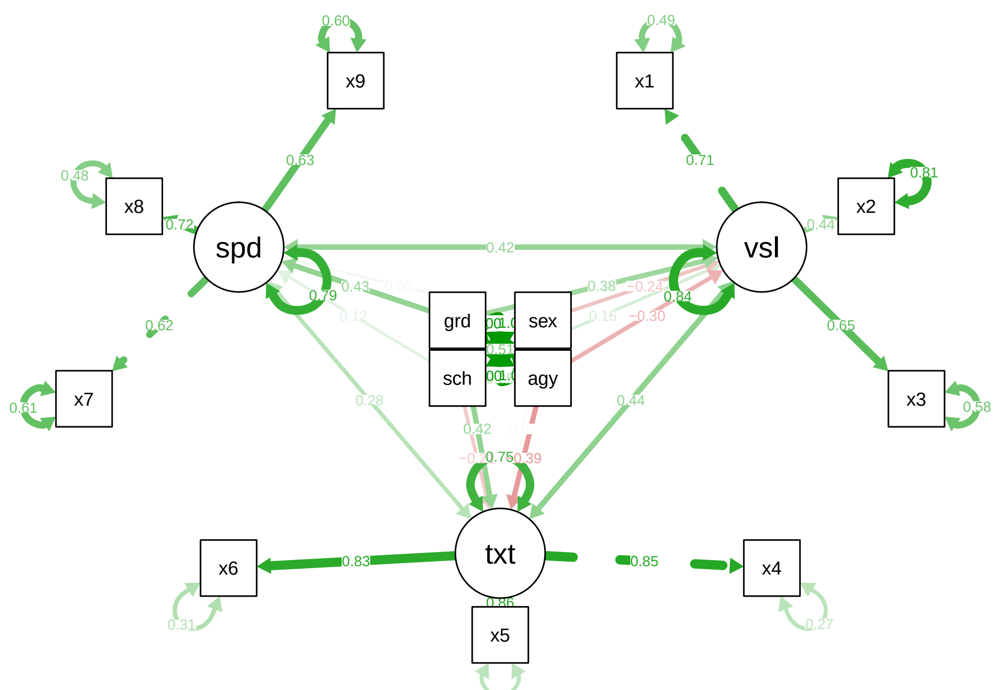
```
## Run the model
fit_ability <- sem(ability_model, data=HolzingerSwineford1939)

## Summarize including fit measures
summary(fit_ability, standardized=T, rsquare=T, fit.measures=T)
```

```
## lavaan (0.6-1) converged normally after  38 iterations
##
##                                                   Used       Total
##   Number of observations                           300         301
##
##   Estimator                                         ML
##   Model Fit Test Statistic                     177.588
##   Degrees of freedom                                48
##   P-value (Chi-square)                           0.000
##
## Model test baseline model:
##
##   Minimum Function Test Statistic             1151.889
##   Degrees of freedom                                72
##   P-value                                        0.000
##
## User model versus baseline model:
##
##   Comparative Fit Index (CFI)                    0.880
##   Tucker-Lewis Index (TLI)                       0.820
##
## Loglikelihood and Information Criteria:
##
##   Loglikelihood user model (H0)              -3655.286
##   Loglikelihood unrestricted model (H1)      -3566.492
##
##   Number of free parameters                         33
##   Akaike (AIC)                                7376.572
##   Bayesian (BIC)                              7498.797
##   Sample-size adjusted Bayesian (BIC)         7394.140
##
## Root Mean Square Error of Approximation:
##
##   RMSEA                                          0.095
##   90 Percent Confidence Interval          0.080  0.110
##   P-value RMSEA <= 0.05                          0.000
##
## Standardized Root Mean Square Residual:
##
```

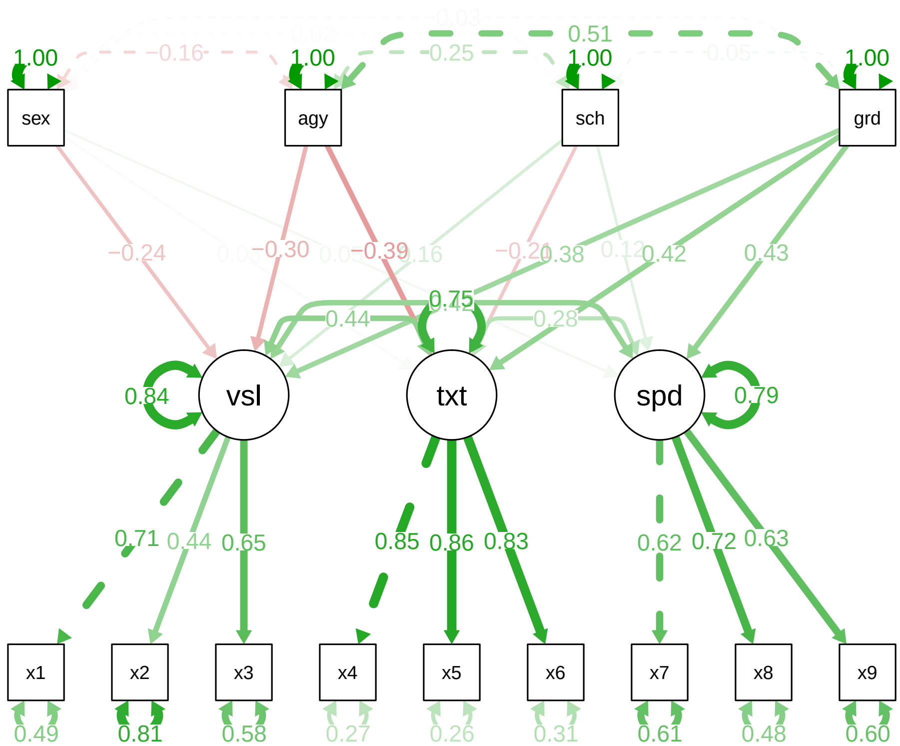
```
##   SRMR                                           0.063
##
## Parameter Estimates:
##
##   Information                                 Expected
##   Information saturated (h1) model          Structured
##   Standard Errors                             Standard
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   visual =~
##     x1                1.000                               0.829    0.711
##     x2                0.618    0.106    5.848    0.000    0.512    0.435
##     x3                0.881    0.118    7.488    0.000    0.730    0.647
##   textual =~
##     x4                1.000                               0.993    0.853
##     x5                1.115    0.064   17.404    0.000    1.107    0.859
##     x6                0.916    0.055   16.788    0.000    0.910    0.831
##   speed =~
##     x7                1.000                               0.679    0.624
##     x8                1.073    0.134    8.002    0.000    0.729    0.722
##     x9                0.934    0.122    7.664    0.000    0.635    0.629
##
## Regressions:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   visual ~
##     sex              -0.392    0.115   -3.391    0.001   -0.472   -0.236
##     ageyr            -0.238    0.066   -3.580    0.000   -0.287   -0.301
##     school            0.271    0.117    2.326    0.020    0.327    0.164
##     grade             0.633    0.136    4.669    0.000    0.763    0.381
##   textual ~
##     sex               0.062    0.110    0.565    0.572    0.062    0.031
##     ageyr            -0.373    0.064   -5.816    0.000   -0.376   -0.395
##     school           -0.419    0.113   -3.708    0.000   -0.422   -0.211
##     grade             0.844    0.129    6.524    0.000    0.851    0.425
##   speed ~
##     sex               0.074    0.088    0.836    0.403    0.109    0.054
##     ageyr             0.001    0.051    0.020    0.984    0.001    0.002
##     school            0.169    0.091    1.860    0.063    0.249    0.125
##     grade             0.580    0.112    5.179    0.000    0.854    0.427
##
## Covariances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##  .visual ~~
##    .textual           0.288    0.058    4.989    0.000    0.441    0.441
##    .speed             0.192    0.047    4.061    0.000    0.417    0.417
##  .textual ~~
##    .speed             0.146    0.043    3.374    0.001    0.279    0.279
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .x1                0.674    0.097    6.927    0.000    0.674    0.495
##    .x2                1.124    0.101   11.086    0.000    1.124    0.811
##    .x3                0.741    0.088    8.420    0.000    0.741    0.582
```

```
##     .x4                0.369    0.046    7.957    0.000    0.369    0.272
##     .x5                0.434    0.056    7.726    0.000    0.434    0.262
##     .x6                0.370    0.043    8.687    0.000    0.370    0.309
##     .x7                0.725    0.078    9.322    0.000    0.725    0.611
##     .x8                0.487    0.067    7.253    0.000    0.487    0.478
##     .x9                0.616    0.067    9.236    0.000    0.616    0.605
##     .visual            0.576    0.110    5.231    0.000    0.838    0.838
##     .textual           0.743    0.087    8.567    0.000    0.754    0.754
##     .speed             0.367    0.075    4.894    0.000    0.795    0.795
##
## R-Square:
##                     Estimate
##     x1                 0.505
##     x2                 0.189
##     x3                 0.418
##     x4                 0.728
##     x5                 0.738
##     x6                 0.691
##     x7                 0.389
##     x8                 0.522
##     x9                 0.395
##     visual             0.162
##     textual            0.246
##     speed              0.205
```

```r
## Visualize your paths (may have to try a few!)
semPaths(fit_ability, 'std', layout='circle')
```



```
semPaths(fit_ability,'std', layout='tree', edge.label.cex =.9, curvePivot=T)
```



Questions?