

P.07 - Measurement Invariance

Mihai A. Constantin

November 15, 2022

Lab Description

During this practical you will learn how to test for various types of measurement invariance when dealing with both cross-sectional (i.e., *Exercise 1*) and longitudinal data (i.e., *Exercise 2*).

For this practical you will need the following packages:

- `lavaan` for structural equation modeling
- `semPlot` for visualizing structural equation models
- `mvtnorm` for generating multivariate normal data
- `GGally` for visualizing multivariate normal data

You can install and load these packages using the following code:

```
# Install packages.
install.packages(c("lavaan", "semPlot", "mvtnorm", "GGally"))

# Load the packages.
library(lavaan)
library(semPlot)
library(mvtnorm)
library(GGally)
```

Exercise 1

This exercise consists of two parts. In the first part, we prepare the data (i.e., in the form of means and covariances). In the second part, you are asked to perform the measurement invariance tests discussed during the lecture. Note that instead of using the entire data for our measurement invariance investigation, we will instead use mean vectors and covariance matrices (i.e., see the *Multiple groups* section from the `lavaan` tutorial).

Part 1. Preparing the data.

In Table 4.3 of Beaujean (2014, p. 66) (i.e., depicted in *Figure 1*) we are presented with the means and covariances for a set of eight random variables. Each random variables is assumed to be normally distributed

and represents an item on the [Wechsler Intelligence Scale for Children-Third Edition](#).

Table 4.3 Covariances and Means for Wechsler Intelligence Scale for Children-Third Edition Subtests.

	Info	Sim	Vocab	Comp	PicComp	PicArr	BlkDsgn	ObjAsmb
Information	9.364	7.777	6.422	5.669	3.048	3.505	3.690	3.640
Similarities	7.777	12.461	8.756	7.445	4.922	4.880	5.440	4.641
Vocabulary	6.422	8.756	10.112	6.797	4.513	4.899	5.220	4.877
Comprehension	5.669	7.445	6.797	8.123	4.116	5.178	3.151	3.568
Picture Completion	3.048	4.922	4.513	4.116	6.200	5.114	3.587	3.819
Picture Arrangement	3.505	4.880	4.899	5.178	5.114	15.603	6.219	5.811
Block Design	3.690	5.440	5.220	3.151	3.587	6.219	11.223	6.501
Object Assembly	3.640	4.641	4.877	3.568	3.819	5.811	6.501	9.797
Subtest Mean	10.090	12.070	10.250	9.960	10.900	11.240	10.300	10.440

(a) Youth with Manic Symptoms ($n = 81$). Data taken from Beaujean et al. (2012, p. 5).

	Info	Sim	Vocab	Comp	PicComp	PicArr	BlkDsgn	ObjAsmb
Information	9.610	5.844	6.324	4.405	4.464	3.478	5.270	4.297
Similarities	5.844	8.410	6.264	4.457	4.547	2.967	4.930	4.594
Vocabulary	6.324	6.264	9.000	5.046	4.512	2.970	4.080	4.356
Comprehension	4.405	4.457	5.046	8.410	3.712	2.871	3.254	3.158
Picture Completion	4.464	4.547	4.512	3.712	10.240	3.802	5.222	4.963
Picture Arrangement	3.478	2.967	2.970	2.871	3.802	10.890	3.590	3.594
Block Design	5.270	4.930	4.080	3.254	5.222	3.590	11.560	6.620
Object Assembly	4.297	4.594	4.356	3.158	4.963	3.594	6.620	10.890
Subtest Mean	10.100	10.300	9.800	10.100	10.100	10.100	9.900	10.200

(b) WISC-III Norming Sample ($n = 200$). Data taken from Wechsler (1991).

Figure 1: Table 4.3 presented in Beaujean (2014).

Together, these items are assumed to measure two latent construct, namely the Verbal-Comprehension (VC) and Visual-Spatial (VS) components of intelligence, as depicted in the model in *Figure 2*.

In our case, we have two groups, (1) one containing youth with manic symptoms (i.e., $N = 81$), and (2) one representing the norming sample (i.e., $N = 200$). To keep things short and readable, we will use the following abbreviations for the variable names:

- `inf` = Information
- `sim` = Similarities
- `voc` = Vocabulary
- `com` = Comprehension
- `p_com` = Picture Completion
- `p_arr` = Picture Arrangement
- `b_des` = Block Design
- `o_ass` = Object Assembly

We start by storing the variable names, means, and covariances.

```
# Variable names.
var_names <- c("inf", "sim", "voc", "com", "p_com", "p_arr", "b_des", "o_ass")

# Means and covariances for group with manic symptoms (i.e., group 1).
```

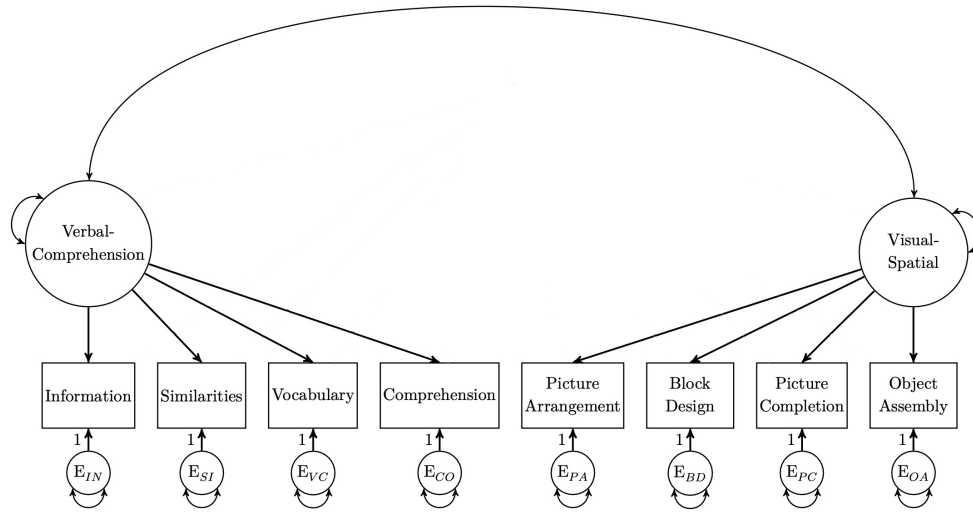


Figure 2: Adaptation of *Figure 4.3* from Beaujean (2014).

```

# Means for group 1.
group_1_means <- c(10.09, 12.07, 10.25, 9.96, 10.90, 11.24, 10.30, 10.44)

# Lower triangle of the covariance matrix for group 1.
group_1_cov_lower <- c(
  9.364, 7.777, 12.461, 6.422, 8.756, 10.112, 5.669,
  7.445, 6.797, 8.123, 3.048, 4.922, 4.513, 4.116,
  6.200, 3.505, 4.880, 4.899, 5.178, 5.114, 15.603,
  3.690, 5.440, 5.220, 3.151, 3.587, 6.219, 11.223,
  3.640, 4.641, 4.877, 3.568, 3.819, 5.811, 6.501, 9.797
)

# Create the entire covariance matrix for group 1.
group_1_cov <- lav_matrix_lower2full(group_1_cov_lower)

# Means and covariances for norming group (i.e., group 2).
# Means for group 2.
group_2_means <- c(10.10, 10.30, 9.80, 10.10, 10.10, 10.10, 9.90, 10.20)

# Lower triangle of the covariance matrix for group 2.
group_2_cov_lower <- c(
  9.610, 5.844, 8.410, 6.324, 6.264, 9.000, 4.405,
  4.457, 5.046, 8.410, 4.464, 4.547, 4.512, 3.712,
  10.240, 3.478, 2.967, 2.970, 2.871, 3.802, 10.890,
  5.270, 4.930, 4.080, 3.254, 5.222, 3.590, 11.560,
  4.297, 4.594, 4.356, 3.158, 4.963, 3.594, 6.620, 10.890
)

# Create the entire covariance matrix for group 2.
group_2_cov <- lav_matrix_lower2full(group_2_cov_lower)

# Add the variable names to the means and covariances for both groups.
# Add the names for the means.
names(group_1_means) <- var_names
names(group_2_means) <- var_names

# Add the names for the covariances.

```

```
rownames(group_1_cov) <- colnames(group_1_cov) <- var_names
rownames(group_2_cov) <- colnames(group_2_cov) <- var_names
```

We can print the the means and covariances for both groups as follows:

```
# Means for group 1.
print(group_1_means)
```

```
##   inf   sim   voc   com p_com p_arr b_des o_ass
## 10.09 12.07 10.25  9.96 10.90 11.24 10.30 10.44
```

```
# Means for group 2.
print(group_2_means)
```

```
##   inf   sim   voc   com p_com p_arr b_des o_ass
## 10.1 10.3  9.8  10.1 10.1 10.1  9.9 10.2
```

```
# Covariances for group 1.
print(group_1_cov)
```

```
##           inf      sim      voc      com p_com p_arr b_des o_ass
## inf   9.364  7.777  6.422  5.669  3.048  3.505  3.690  3.640
## sim   7.777 12.461  8.756  7.445  4.922  4.880  5.440  4.641
## voc   6.422  8.756 10.112  6.797  4.513  4.899  5.220  4.877
## com   5.669  7.445  6.797  8.123  4.116  5.178  3.151  3.568
## p_com 3.048  4.922  4.513  4.116  6.200  5.114  3.587  3.819
## p_arr 3.505  4.880  4.899  5.178  5.114 15.603  6.219  5.811
## b_des 3.690  5.440  5.220  3.151  3.587  6.219 11.223  6.501
## o_ass 3.640  4.641  4.877  3.568  3.819  5.811  6.501  9.797
```

```
# Covariances for group 2.
print(group_2_cov)
```

```
##           inf      sim      voc      com p_com p_arr b_des o_ass
## inf   9.610  5.844  6.324  4.405  4.464  3.478  5.270  4.297
## sim   5.844  8.410  6.264  4.457  4.547  2.967  4.930  4.594
## voc   6.324  6.264  9.000  5.046  4.512  2.970  4.080  4.356
## com   4.405  4.457  5.046  8.410  3.712  2.871  3.254  3.158
## p_com 4.464  4.547  4.512  3.712 10.240  3.802  5.222  4.963
## p_arr 3.478  2.967  2.970  2.871  3.802 10.890  3.590  3.594
## b_des 5.270  4.930  4.080  3.254  5.222  3.590 11.560  6.620
## o_ass 4.297  4.594  4.356  3.158  4.963  3.594  6.620 10.890
```

Finally, as we can see in the documentation of `lavaan` for arguments `sample.cov`, `sample.mean`, and `sample.nobs`, for multiple group analysis we need to specify a list of mean vectors and covariance matrices:

`sample.cov`: For a multiple group analysis, a list with a variance-covariance matrix for each group.

`sample.mean`: For a multiple group analysis, a list with a mean vector for each group.

`sample.nobs`: For a multiple group analysis, a list or a vector with the number of observations for each group.

We can create the required lists as follows:

```
# Combine the mean vectors into a single list.
means <- list(
  group_1 = group_1_means,
  group_2 = group_2_means
)
```

```
# Combine the covariance matrices into a single list.
covariances <- list(
  group_1 = group_1_cov,
  group_2 = group_2_cov
)

# Combine the sample sizes into a single list.
samples <- list(
  group_1 = 81,
  group_2 = 200
)
```

Specify which fit measures we are interested in:

```
# Fit indices to print.
fit_indices <- c("chisq", "df", "pvalue", "cfi", "tli", "rmsea", "rmsea.pvalue", "srmr")
```

Part 2. Measurement invariance investigation.

a. Write down the model syntax for the model in *Figure 2*.

```
# Model syntax.
model_ex_1 <- "
  # Measurement part.
  verbal =~ inf + sim + voc + com
  visual =~ p_com + p_arr + b_des + o_ass

  # Latent covariance.
  verbal ~~ visual
"
```

b. Manually fit the model in *Figure 2* for each group and allow the means of the observed variables to enter the model. Report and interpret the model fit.

To allow the observed means to enter the model, we need to set `meanstructure = TRUE`, in the `lavaan` function `sem` or `cfa`. First, we fit the model for the first group.

```
# Fit the model for group 1.
model_ex_1_group_1_fit <- sem(
  model_ex_1,
  sample.cov = group_1_cov,
  sample.nobs = 81,
  sample.mean = group_1_means,
  meanstructure = TRUE
)

# Model summary.
summary(model_ex_1_group_1_fit, standardized = TRUE)
```

```
## lavaan 0.6-12 ended normally after 42 iterations
##
## Estimator ML
## Optimization method NLMINB
## Number of model parameters 25
##
## Number of observations 81
##
## Model Test User Model:
##
## Test statistic 29.169
```

```

## Degrees of freedom          19
## P-value (Chi-square)       0.063
##
## Parameter Estimates:
##
## Standard errors            Standard
## Information                Expected
## Information saturated (h1) model Structured
##
## Latent Variables:
##      Estimate Std.Err z-value P(>|z|) Std.lv Std.all
## verbal =~
##   inf          1.000
##   sim          1.330    0.153    8.687    0.000    3.121    0.890
##   voc          1.189    0.138    8.613    0.000    2.791    0.883
##   com          1.015    0.125    8.129    0.000    2.382    0.841
## visual =~
##   p_com        1.000
##   p_arr        1.437    0.274    5.246    0.000    2.570    0.655
##   b_des        1.322    0.234    5.641    0.000    2.364    0.710
##   o_ass        1.285    0.220    5.830    0.000    2.297    0.738
##
## Covariances:
##      Estimate Std.Err z-value P(>|z|) Std.lv Std.all
## verbal ~~
##   visual       3.086    0.772    3.997    0.000    0.735    0.735
##
## Intercepts:
##      Estimate Std.Err z-value P(>|z|) Std.lv Std.all
##   .inf        10.090    0.338    29.861    0.000    10.090    3.318
##   .sim        12.070    0.390    30.965    0.000    12.070    3.441
##   .voc        10.250    0.351    29.191    0.000    10.250    3.243
##   .com         9.960    0.315    31.648    0.000     9.960    3.516
##   .p_com       10.900    0.275    39.643    0.000    10.900    4.405
##   .p_arr       11.240    0.436    25.769    0.000    11.240    2.863
##   .b_des       10.300    0.370    27.843    0.000    10.300    3.094
##   .o_ass       10.440    0.346    30.206    0.000    10.440    3.356
##   verbal        0.000
##   visual        0.000
##
## Variances:
##      Estimate Std.Err z-value P(>|z|) Std.lv Std.all
##   .inf         3.742    0.673    5.564    0.000    3.742    0.405
##   .sim         2.564    0.605    4.237    0.000    2.564    0.208
##   .voc         2.200    0.502    4.379    0.000    2.200    0.220
##   .com         2.348    0.467    5.027    0.000    2.348    0.293
##   .p_com       2.926    0.592    4.945    0.000    2.926    0.478
##   .p_arr       8.806    1.631    5.398    0.000    8.806    0.571
##   .b_des       5.497    1.089    5.046    0.000    5.497    0.496
##   .o_ass       4.399    0.916    4.804    0.000    4.399    0.455
##   verbal       5.507    1.369    4.024    0.000    1.000    1.000
##   visual       3.198    0.924    3.462    0.001    1.000    1.000

# Fit measures.
fitMeasures(model_ex_1_group_1_fit, fit.measures = fit_indices)

##      chisq      df      pvalue      cfi      tli      rmsea rmsea.pvalue
##      29.169    19.000      0.063     0.971     0.958     0.081      0.184
##      srmr
##      0.047

```

Then, we fit the model for the second group.

```
# Fit the model for group 1.
model_ex_1_group_2_fit <- sem(
  model_ex_1,
  sample.cov = group_2_cov,
  sample.nobs = 200,
  sample.mean = group_2_means,
  meanstructure = TRUE
)

# Model summary.
summary(model_ex_1_group_2_fit, standardized = TRUE)
```

```
## lavaan 0.6-12 ended normally after 45 iterations
##
##      Estimator                      ML
##      Optimization method          NLMINB
##      Number of model parameters    25
##
##      Number of observations        200
##
## Model Test User Model:
##
##      Test statistic                24.211
##      Degrees of freedom            19
##      P-value (Chi-square)          0.188
##
## Parameter Estimates:
##
##      Standard errors              Standard
##      Information                  Expected
##      Information saturated (h1) model Structured
##
## Latent Variables:
##      Estimate Std.Err z-value P(>|z|) Std.lv Std.all
##      verbal =~
##      .inf      1.000      0.079 12.607 0.000 2.440 0.789
##      .sim      0.997      0.079 12.607 0.000 2.433 0.841
##      .voc      1.045      0.082 12.778 0.000 2.550 0.852
##      .com      0.768      0.083 9.301 0.000 1.874 0.648
##      visual =~
##      .p_com     1.000      0.122 5.854 0.000 2.182 0.684
##      .p_arr     0.715      0.135 8.542 0.000 1.561 0.474
##      .b_des     1.149      0.130 8.464 0.000 2.507 0.739
##      .o_ass     1.100      0.130 8.464 0.000 2.401 0.730
##
## Covariances:
##      Estimate Std.Err z-value P(>|z|) Std.lv Std.all
##      verbal ~~
##      visual     4.103      0.661 6.204 0.000 0.771 0.771
##
## Intercepts:
##      Estimate Std.Err z-value P(>|z|) Std.lv Std.all
##      .inf      10.100      0.219 46.192 0.000 10.100 3.266
##      .sim      10.300      0.205 50.355 0.000 10.300 3.561
##      .voc       9.800      0.212 46.314 0.000 9.800 3.275
##      .com      10.100      0.205 49.377 0.000 10.100 3.491
##      .p_com     10.100      0.226 44.748 0.000 10.100 3.164
##      .p_arr     10.100      0.233 43.392 0.000 10.100 3.068
##      .b_des      9.900      0.240 41.282 0.000 9.900 2.919
```

```
##      .o_ass      10.200    0.233   43.822    0.000   10.200    3.099
##      verbal      0.000                      0.000    0.000
##      visual      0.000                      0.000    0.000
##
## Variances:
##      Estimate Std.Err z-value P(>|z|) Std.lv Std.all
##      .inf      3.609   0.455   7.928   0.000   3.609   0.377
##      .sim      2.450   0.354   6.925   0.000   2.450   0.293
##      .voc      2.453   0.370   6.635   0.000   2.453   0.274
##      .com      4.857   0.533   9.114   0.000   4.857   0.580
##      .p_com     5.426   0.675   8.034   0.000   5.426   0.533
##      .p_arr     8.398   0.897   9.364   0.000   8.398   0.775
##      .b_des     5.215   0.716   7.279   0.000   5.215   0.453
##      .o_ass     5.069   0.682   7.434   0.000   5.069   0.468
##      verbal     5.953   0.928   6.416   0.000   1.000   1.000
##      visual     4.763   0.952   5.006   0.000   1.000   1.000
```

```
# Fit measures.
fitMeasures(model_ex_1_group_2_fit, fit.measures = fit_indices)
```

```
##      chisq      df      pvalue      cfi      tli      rmsea rmsea.pvalue
##      24.211    19.000      0.188      0.992      0.989      0.037      0.662
##      srmr
##      0.029
```

c. Investigate *configural* measurement invariance. Report and interpret the model fit.

In fact, *we already investigated configural measurement invariance* at point (b) by manually fitting the model in each group and checking the global fit and whether the loadings are significant for both groups. This time, instead of manually fitting the model to each group, we will pass the list of mean vectors, covariance matrices, and sample sizes to *lavaan*, which will automate the fitting for us.

```
# Fit the model to both groups.
model_ex_1_configural_fit <- sem(
  model_ex_1,
  sample.cov = covariances,
  sample.nobs = samples,
  sample.mean = means
)

# Model summary.
summary(model_ex_1_configural_fit, standardized = TRUE)
```

```
## lavaan 0.6-12 ended normally after 93 iterations
##
## Estimator ML
## Optimization method NLMINB
## Number of model parameters 50
##
## Number of observations per group:
## group_1 81
## group_2 200
##
## Model Test User Model:
##
## Test statistic 53.380
## Degrees of freedom 38
## P-value (Chi-square) 0.050
## Test statistic for each group:
## group_1 29.169
## group_2 24.211
```



```

##
## Parameter Estimates:
##
## Standard errors          Standard
## Information              Expected
## Information saturated (h1) model  Structured
##
##
## Group 1 [group_1]:
##
## Latent Variables:
##      Estimate Std.Err z-value P(>|z|) Std.lv Std.all
## verbal =~
##   inf          1.000
##   sim          1.330    0.153    8.687    0.000    3.121    0.890
##   voc          1.189    0.138    8.613    0.000    2.791    0.883
##   com          1.015    0.125    8.129    0.000    2.382    0.841
## visual =~
##   p_com          1.000
##   p_arr          1.437    0.274    5.246    0.000    2.570    0.655
##   b_des          1.322    0.234    5.641    0.000    2.364    0.710
##   o_ass          1.285    0.220    5.830    0.000    2.297    0.738
##
## Covariances:
##      Estimate Std.Err z-value P(>|z|) Std.lv Std.all
## verbal ~~
##   visual          3.086    0.772    3.997    0.000    0.735    0.735
##
## Intercepts:
##      Estimate Std.Err z-value P(>|z|) Std.lv Std.all
##   .inf          10.090    0.338    29.861    0.000    10.090    3.318
##   .sim          12.070    0.390    30.965    0.000    12.070    3.441
##   .voc          10.250    0.351    29.191    0.000    10.250    3.243
##   .com          9.960    0.315    31.648    0.000    9.960    3.516
##   .p_com        10.900    0.275    39.643    0.000    10.900    4.405
##   .p_arr        11.240    0.436    25.769    0.000    11.240    2.863
##   .b_des        10.300    0.370    27.843    0.000    10.300    3.094
##   .o_ass        10.440    0.346    30.206    0.000    10.440    3.356
##   verbal         0.000
##   visual         0.000
##
## Variances:
##      Estimate Std.Err z-value P(>|z|) Std.lv Std.all
##   .inf          3.742    0.673    5.564    0.000    3.742    0.405
##   .sim          2.564    0.605    4.237    0.000    2.564    0.208
##   .voc          2.200    0.502    4.379    0.000    2.200    0.220
##   .com          2.348    0.467    5.027    0.000    2.348    0.293
##   .p_com        2.926    0.592    4.945    0.000    2.926    0.478
##   .p_arr        8.806    1.631    5.398    0.000    8.806    0.571
##   .b_des        5.497    1.089    5.046    0.000    5.497    0.496
##   .o_ass        4.399    0.916    4.804    0.000    4.399    0.455
##   verbal        5.507    1.369    4.024    0.000    1.000    1.000
##   visual        3.198    0.924    3.462    0.001    1.000    1.000
##
##
## Group 2 [group_2]:
##
## Latent Variables:
##      Estimate Std.Err z-value P(>|z|) Std.lv Std.all
## verbal =~
##   inf          1.000
##              2.440    0.789

```

```
##      sim            0.997    0.079   12.607    0.000    2.433    0.841
##      voc            1.045    0.082   12.778    0.000    2.550    0.852
##      com            0.768    0.083    9.301    0.000    1.874    0.648
##      visual =~
##      p_com          1.000
##      p_arr          0.715    0.122    5.854    0.000    1.561    0.474
##      b_des          1.149    0.135    8.542    0.000    2.507    0.739
##      o_ass          1.100    0.130    8.464    0.000    2.401    0.730
##
## Covariances:
##      Estimate Std.Err z-value P(>|z|) Std.lv Std.all
##      verbal ~~
##      visual      4.103    0.661    6.204    0.000    0.771    0.771
##
## Intercepts:
##      Estimate Std.Err z-value P(>|z|) Std.lv Std.all
##      .inf      10.100    0.219   46.192    0.000   10.100    3.266
##      .sim       10.300    0.205   50.355    0.000   10.300    3.561
##      .voc       9.800    0.212   46.314    0.000    9.800    3.275
##      .com       10.100    0.205   49.377    0.000   10.100    3.491
##      .p_com     10.100    0.226   44.748    0.000   10.100    3.164
##      .p_arr     10.100    0.233   43.392    0.000   10.100    3.068
##      .b_des     9.900    0.240   41.282    0.000    9.900    2.919
##      .o_ass     10.200    0.233   43.822    0.000   10.200    3.099
##      verbal      0.000
##      visual      0.000
##
## Variances:
##      Estimate Std.Err z-value P(>|z|) Std.lv Std.all
##      .inf       3.609    0.455    7.928    0.000    3.609    0.377
##      .sim       2.450    0.354    6.925    0.000    2.450    0.293
##      .voc       2.453    0.370    6.635    0.000    2.453    0.274
##      .com       4.857    0.533    9.114    0.000    4.857    0.580
##      .p_com     5.426    0.675    8.034    0.000    5.426    0.533
##      .p_arr     8.398    0.897    9.364    0.000    8.398    0.775
##      .b_des     5.215    0.716    7.279    0.000    5.215    0.453
##      .o_ass     5.069    0.682    7.434    0.000    5.069    0.468
##      verbal     5.953    0.928    6.416    0.000    1.000    1.000
##      visual     4.763    0.952    5.006    0.000    1.000    1.000
```

```
# Fit measures.
fitMeasures(model_ex_1_configural_fit, fit.measures = fit_indices)
```

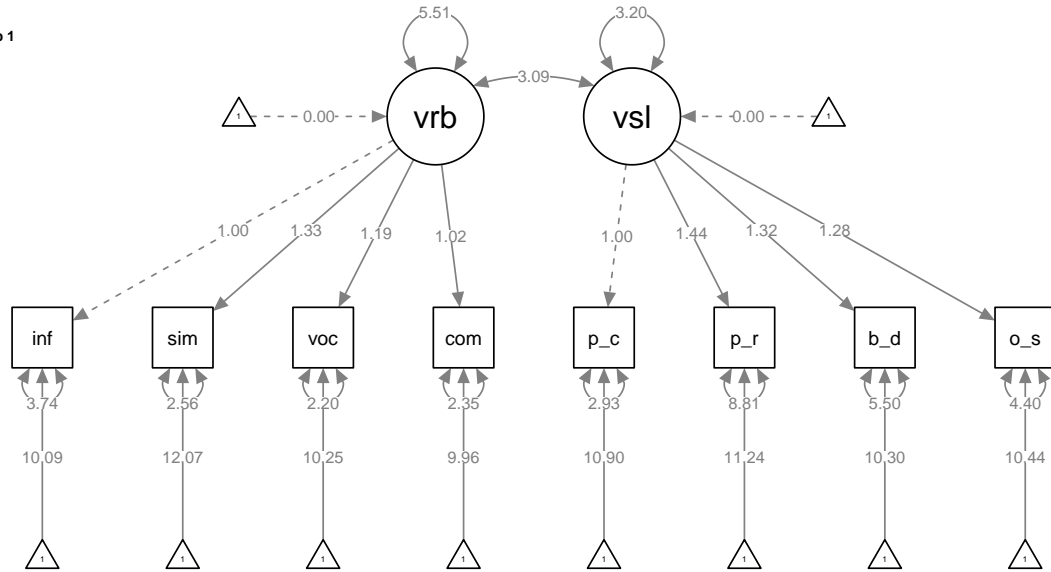
```
##      chisq      df      pvalue      cfi      tli      rmsea rmsea.pvalue
##      53.380    38.000      0.050     0.985     0.978     0.054     0.402
##      srmr
##      0.034
```

We see that the model has an adequate fit for each group. Before we continue, we can also plot the model for each group.

```
# Save the model plots for each groups as a list with two elements.
plots_configural <- semPaths(
  model_ex_1_configural_fit,
  what = "paths",
  whatLabels = "est",
  DoNotPlot = TRUE,
  ask = FALSE,
  title = FALSE
)
```

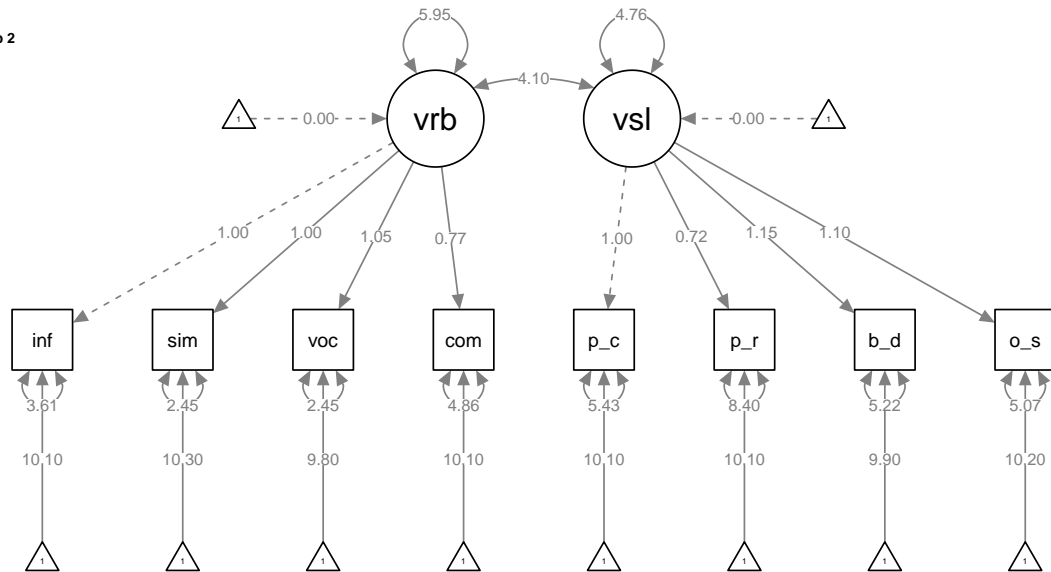
```
# Plot the model for the first group.
plot(plots_configural[[1]])
title("Configural MI | Group 1", adj = 0)
```

Configural MI | Group 1



```
# Plot the model for the second group.
plot(plots_configural[[2]])
title("Configural MI | Group 2", adj = 0)
```

Configural MI | Group 2



d. Investigate *weak* or *metric* measurement invariance. Report and interpret the model fit.

To investigate metric measurement invariance, we need to constrain the loadings to be equal in both groups. To achieve this, we can use the `group.equal` argument in `lavaan`.

```
# Fit the model.
model_ex_1_weak_fit <- sem(
  model_ex_1,
  sample.cov = covariances,
```

```

    sample.nobs = samples,
    sample.mean = means,
    group.equal = c("loadings")
)

```

```

# Model summary.
summary(model_ex_1_weak_fit, standardized = TRUE)

```

```

## lavaan 0.6-12 ended normally after 94 iterations
##
##      Estimator                      ML
##      Optimization method          NLMINB
##      Number of model parameters      50
##      Number of equality constraints    6
##
##      Number of observations per group:
##      group_1                        81
##      group_2                       200
##
## Model Test User Model:
##
##      Test statistic                65.992
##      Degrees of freedom             44
##      P-value (Chi-square)           0.018
##      Test statistic for each group:
##      group_1                       37.832
##      group_2                       28.160
##
## Parameter Estimates:
##
##      Standard errors                Standard
##      Information                    Expected
##      Information saturated (h1) model Structured
##
##
## Group 1 [group_1]:
##
## Latent Variables:
##      Estimate Std.Err z-value P(>|z|) Std.lv Std.all
##      verbal =~
##      inf      1.000
##      sim      1.105    0.073   15.193   0.000   2.908   0.867
##      voc      1.094    0.071   15.298   0.000   2.877   0.893
##      com      0.864    0.068   12.720   0.000   2.273   0.825
##      visual =~
##      p_com     1.000
##      p_arr     0.888    0.118    7.554   0.000   1.739   0.480
##      b_des     1.217    0.121   10.089   0.000   2.383   0.711
##      o_ass     1.174    0.116   10.148   0.000   2.298   0.738
##
## Covariances:

```

```

##               Estimate Std.Err z-value P(>|z|) Std.lv Std.all
##  verbal ~~
##    visual      3.865    0.845    4.571    0.000    0.750    0.750
##
## Intercepts:
##               Estimate Std.Err z-value P(>|z|) Std.lv Std.all
##    .inf          10.090    0.363    27.815    0.000    10.090    3.091
##    .sim           12.070    0.373    32.397    0.000    12.070    3.600
##    .voc           10.250    0.358    28.639    0.000    10.250    3.182
##    .com            9.960    0.306    32.548    0.000     9.960    3.616
##    .p_com         10.900    0.287    37.996    0.000    10.900    4.222
##    .p_arr         11.240    0.403    27.915    0.000    11.240    3.102
##    .b_des         10.300    0.373    27.646    0.000    10.300    3.072
##    .o_ass         10.440    0.346    30.170    0.000    10.440    3.352
##    verbal          0.000                0.000    0.000
##    visual          0.000                0.000    0.000
##
## Variances:
##               Estimate Std.Err z-value P(>|z|) Std.lv Std.all
##    .inf            3.739    0.694     5.386    0.000     3.739    0.351
##    .sim            2.787    0.589     4.733    0.000     2.787    0.248
##    .voc            2.097    0.494     4.242    0.000     2.097    0.202
##    .com            2.419    0.464     5.217    0.000     2.419    0.319
##    .p_com          2.831    0.599     4.725    0.000     2.831    0.425
##    .p_arr          10.109    1.678     6.024    0.000    10.109    0.770
##    .b_des           5.564    1.079     5.158    0.000     5.564    0.495
##    .o_ass           4.417    0.895     4.936    0.000     4.417    0.455
##    verbal           6.920    1.362     5.081    0.000     1.000    1.000
##    visual           3.835    0.877     4.371    0.000     1.000    1.000
##
##
## Group 2 [group_2]:
##
## Latent Variables:
##               Estimate Std.Err z-value P(>|z|) Std.lv Std.all
##    verbal =~
##      inf          1.000                2.286    0.762
##      sim    (.p2.)  1.105    0.073    15.193    0.000    2.527    0.855
##      voc     (.p3.)  1.094    0.071    15.298    0.000    2.500    0.844
##      com     (.p4.)  0.864    0.068    12.720    0.000    1.975    0.669
##    visual =~
##      p_com          1.000                2.047    0.655
##      p_arr    (.p6.)  0.888    0.118     7.554    0.000    1.817    0.534
##      b_des    (.p7.)  1.217    0.121    10.089    0.000    2.491    0.736
##      o_ass    (.p8.)  1.174    0.116    10.148    0.000    2.402    0.730
##
## Covariances:
##               Estimate Std.Err z-value P(>|z|) Std.lv Std.all
##    verbal ~~
##      visual      3.607    0.558     6.461    0.000    0.771    0.771
##

```

```
## Intercepts:
##           Estimate Std.Err z-value P(>|z|) Std.lv Std.all
##   .inf          10.100   0.212  47.646   0.000   10.100   3.369
##   .sim          10.300   0.209  49.298   0.000   10.300   3.486
##   .voc           9.800   0.209  46.810   0.000    9.800   3.310
##   .com          10.100   0.209  48.367   0.000   10.100   3.420
##   .p_com        10.100   0.221  45.728   0.000   10.100   3.233
##   .p_arr        10.100   0.241  41.952   0.000   10.100   2.966
##   .b_des         9.900   0.239  41.391   0.000    9.900   2.927
##   .o_ass        10.200   0.233  43.843   0.000   10.200   3.100
##   verbal         0.000                0.000   0.000
##   visual         0.000                0.000   0.000
##
## Variances:
##           Estimate Std.Err z-value P(>|z|) Std.lv Std.all
##   .inf           3.762   0.452   8.329   0.000    3.762   0.419
##   .sim           2.346   0.352   6.660   0.000    2.346   0.269
##   .voc           2.516   0.362   6.953   0.000    2.516   0.287
##   .com           4.821   0.532   9.056   0.000    4.821   0.553
##   .p_com         5.568   0.662   8.415   0.000    5.568   0.571
##   .p_arr         8.290   0.905   9.155   0.000    8.290   0.715
##   .b_des         5.238   0.705   7.432   0.000    5.238   0.458
##   .o_ass         5.055   0.671   7.535   0.000    5.055   0.467
##   verbal         5.225   0.770   6.783   0.000    1.000   1.000
##   visual         4.189   0.777   5.389   0.000    1.000   1.000
```

```
# Fit measures.
fitMeasures(model_ex_1_weak_fit, fit.measures = fit_indices)
```

```
##           chisq      df      pvalue      cfi      tli      rmsea rmsea.pvalue
##           65.992    44.000        0.018      0.979      0.973      0.060        0.281
##           srmr
##           0.055
```

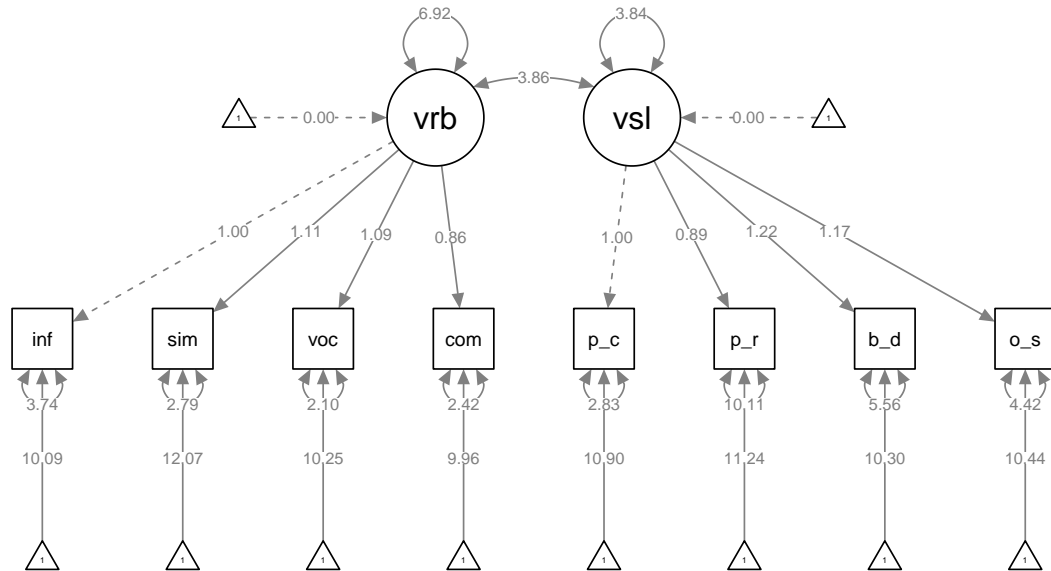
```
# Save the model plots for each groups as a list with two elements.
```

```
plots_weak <- semPaths(
  model_ex_1_weak_fit,
  what = "paths",
  whatLabels = "est",
  DoNotPlot = TRUE,
  ask = FALSE,
  title = FALSE
)
```

```
# Plot the model for the first group.
```

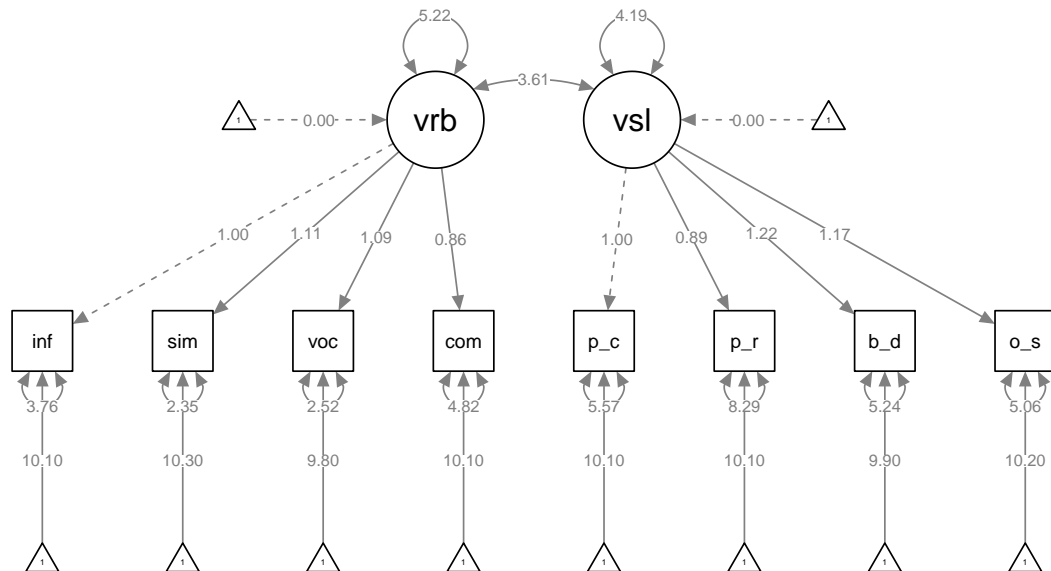
```
plot(plots_weak[[1]])
title("Weak MI | Group 1", adj = 0)
```

Weak MI | Group 1



```
# Plot the model for the second group.
plot(plots_weak[[2]])
title("Weak MI | Group 2", adj = 0)
```

Weak MI | Group 2



- e. Check if the the model fit at point (d) does not significantly worsen the fit compared to the model fit at point (c).

Since these models are nested, can use the `anova()` function in R to perform a *LRT*.

```
# Perform LRT.
anova(model_ex_1_weak_fit, model_ex_1_configural_fit)

## Chi-Squared Difference Test
##
##               Df  AIC   BIC  Chisq Chisq diff Df diff Pr(>Chisq)
## model_ex_1_configural_fit 38 10583 10765 53.380
## model_ex_1_weak_fit      44 10584 10744 65.992    12.613     6  0.04961 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Our findings show support for weak measurement invariance. We see a χ^2 difference value of 12.613 with a p -value = 0.049. As discussed during the lecture, the χ^2 difference test is often significant for the tiniest differences. A recommended practice is to also look at the decrease in *CFI* and increase in *RMSEA* when moving to a higher level of measurement invariance. For two groups, the decrease in *CFI* and increase in *RMSEA* should not exceed 0.01, which we see is the case.

f. Investigate *strong* or *scalar* measurement invariance. Report and interpret the model fit.

To investigate scalar measurement invariance, on top of constraining the loadings to be equal in both groups, we also need to constrain the intercepts. To achieve this, we can, again, use the `group.equal` argument in `lavaan`.

```
# Fit the model.
model_ex_1_strong_fit <- sem(
  model_ex_1,
  sample.cov = covariances,
  sample.nobs = samples,
  sample.mean = means,
  group.equal = c("loadings", "intercepts")
)

# Model summary.
summary(model_ex_1_strong_fit, standardized = TRUE)
```

```
## lavaan 0.6-12 ended normally after 105 iterations
##
##      Estimator                      ML
##      Optimization method          NLMINB
##      Number of model parameters      52
##      Number of equality constraints    14
##
##      Number of observations per group:
##      group_1                        81
##      group_2                       200
##
## Model Test User Model:
##
##      Test statistic                  109.088
##      Degrees of freedom              50
##      P-value (Chi-square)            0.000
##      Test statistic for each group:
##      group_1                        70.405
##      group_2                        38.684
##
## Parameter Estimates:
##
##      Standard errors                Standard
##      Information                    Expected
##      Information saturated (h1) model Structured
##
##
```



```

## Group 1 [group_1]:
##
## Latent Variables:
##      Estimate Std.Err z-value P(>|z|) Std.lv Std.all
## verbal =~
##      inf      1.000
##      sim      (.p2.) 1.118 0.077 14.588 0.000 2.891 0.818
##      voc      (.p3.) 1.110 0.073 15.103 0.000 2.869 0.900
##      com      (.p4.) 0.858 0.070 12.321 0.000 2.218 0.809
## visual =~
##      p_com      1.000
##      p_arr      (.p6.) 0.892 0.116 7.722 0.000 1.767 0.482
##      b_des      (.p7.) 1.188 0.117 10.188 0.000 2.353 0.703
##      o_ass      (.p8.) 1.140 0.112 10.222 0.000 2.258 0.726
##
## Covariances:
##      Estimate Std.Err z-value P(>|z|) Std.lv Std.all
## verbal ~~
##      visual      3.894 0.847 4.595 0.000 0.761 0.761
##
## Intercepts:
##      Estimate Std.Err z-value P(>|z|) Std.lv Std.all
##      .inf      (.20.) 10.477 0.321 32.647 0.000 10.477 3.205
##      .sim      (.21.) 11.115 0.351 31.644 0.000 11.115 3.144
##      .voc      (.22.) 10.338 0.341 30.287 0.000 10.338 3.242
##      .com      (.23.) 10.295 0.279 36.925 0.000 10.295 3.755
##      .p_com     (.24.) 10.749 0.265 40.565 0.000 10.749 4.139
##      .p_arr     (.25.) 10.736 0.283 37.941 0.000 10.736 2.928
##      .b_des     (.26.) 10.466 0.321 32.646 0.000 10.466 3.126
##      .o_ass     (.27.) 10.690 0.305 35.032 0.000 10.690 3.438
##      verbal      0.000
##      visual      0.000
##
## Variances:
##      Estimate Std.Err z-value P(>|z|) Std.lv Std.all
##      .inf      4.007 0.743 5.391 0.000 4.007 0.375
##      .sim      4.146 0.800 5.184 0.000 4.146 0.332
##      .voc      1.936 0.501 3.861 0.000 1.936 0.190
##      .com      2.595 0.496 5.237 0.000 2.595 0.345
##      .p_com     2.823 0.605 4.667 0.000 2.823 0.418
##      .p_arr     10.327 1.716 6.017 0.000 10.327 0.768
##      .b_des     5.677 1.091 5.204 0.000 5.677 0.506
##      .o_ass     4.570 0.909 5.025 0.000 4.570 0.473
##      verbal     6.682 1.333 5.011 0.000 1.000 1.000
##      visual     3.923 0.893 4.391 0.000 1.000 1.000
##
##
## Group 2 [group_2]:
##
## Latent Variables:
##      Estimate Std.Err z-value P(>|z|) Std.lv Std.all

```

```
## verbal =~
##   inf          1.000          2.266  0.758
##   sim   (.p2.)  1.118  0.077  14.588  0.000  2.533  0.851
##   voc   (.p3.)  1.110  0.073  15.103  0.000  2.515  0.847
##   com   (.p4.)  0.858  0.070  12.321  0.000  1.944  0.660
## visual =~
##   p_com        1.000          2.079  0.662
##   p_arr   (.p6.)  0.892  0.116   7.722  0.000  1.854  0.541
##   b_des   (.p7.)  1.188  0.117  10.188  0.000  2.469  0.732
##   o_ass   (.p8.)  1.140  0.112  10.222  0.000  2.370  0.723
##
## Covariances:
##           Estimate Std.Err z-value P(>|z|) Std.lv Std.all
## verbal ~~
##   visual      3.639   0.562   6.478   0.000   0.773   0.773
##
## Intercepts:
##           Estimate Std.Err z-value P(>|z|) Std.lv Std.all
##   .inf   (.20.)  10.477   0.321  32.647   0.000  10.477   3.505
##   .sim   (.21.)  11.115   0.351  31.644   0.000  11.115   3.735
##   .voc   (.22.)  10.338   0.341  30.287   0.000  10.338   3.482
##   .com   (.23.)  10.295   0.279  36.925   0.000  10.295   3.495
##   .p_com (.24.)  10.749   0.265  40.565   0.000  10.749   3.424
##   .p_arr (.25.)  10.736   0.283  37.941   0.000  10.736   3.134
##   .b_des (.26.)  10.466   0.321  32.646   0.000  10.466   3.101
##   .o_ass (.27.)  10.690   0.305  35.032   0.000  10.690   3.261
##   verbal      -0.525   0.348  -1.510   0.131  -0.232  -0.232
##   visual      -0.529   0.301  -1.759   0.079  -0.255  -0.255
##
## Variances:
##           Estimate Std.Err z-value P(>|z|) Std.lv Std.all
##   .inf          3.799   0.455   8.352   0.000   3.799   0.425
##   .sim          2.440   0.364   6.704   0.000   2.440   0.275
##   .voc          2.489   0.364   6.836   0.000   2.489   0.282
##   .com          4.896   0.539   9.087   0.000   4.896   0.564
##   .p_com        5.534   0.663   8.346   0.000   5.534   0.562
##   .p_arr        8.301   0.911   9.117   0.000   8.301   0.707
##   .b_des        5.293   0.706   7.495   0.000   5.293   0.465
##   .o_ass        5.128   0.672   7.628   0.000   5.128   0.477
##   verbal        5.133   0.766   6.698   0.000   1.000   1.000
##   visual        4.321   0.791   5.459   0.000   1.000   1.000
```

```
# Fit measures.
fitMeasures(model_ex_1_strong_fit, fit.measures = fit_indices)
```

```
##      chisq      df      pvalue      cfi      tli      rmsea rmsea.pvalue
##    109.088    50.000      0.000     0.942     0.935     0.092      0.003
##      srmr
##      0.064
```

```
# Save the model plots for each groups as a list with two elements.
plots_strong <- semPaths(
  model_ex_1_strong_fit,
```

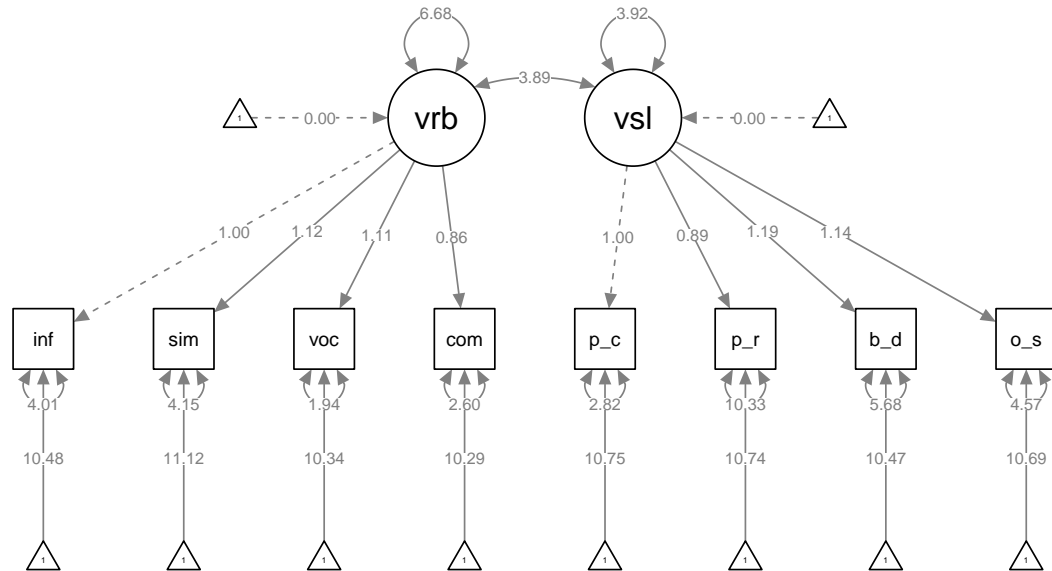
```

what = "paths",
whatLabels = "est",
DoNotPlot = TRUE,
ask = FALSE,
title = FALSE
)

# Plot the model for the first group.
plot(plots_strong[[1]])
title("Strong MI | Group 1", adj = 0)

```

Strong MI | Group 1

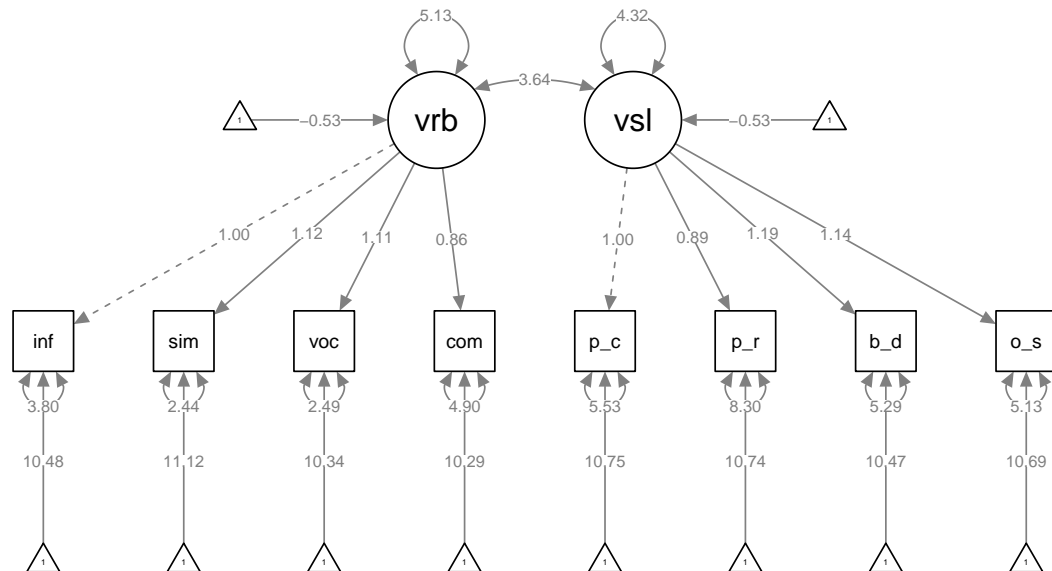


```

# Plot the model for the second group.
plot(plots_strong[[2]])
title("Strong MI | Group 2", adj = 0)

```

Strong MI | Group 2



g. Check if the the model fit at point (f) does not significantly worsen the fit compared to the model fit at

point (*d*).

Since these models are nested, can use the `anova()` function in R to perform a LRT.

```
# Perform LRT.
anova(model_ex_1_strong_fit, model_ex_1_weak_fit)

## Chi-Squared Difference Test
##
##              Df  AIC   BIC   Chisq Chisq diff Df diff Pr(>Chisq)
## model_ex_1_weak_fit  44 10584 10744  65.992
## model_ex_1_strong_fit 50 10615 10753 109.088    43.096      6 1.117e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Our findings do not show support for strong measurement invariance (i.e., the *strong* measurement invariance model fits significantly worse compared to the *weak* measurement invariance model). We obtain a χ^2 difference value of 43.096 with a p -value < 0.001 . Also, the rules of thumbs regarding the decrease in *CFI* and increase in *RMSEA* show that we do not have support for strong measurement invariance.

- h. Based on the findings above, should you continue with investigating strict measurement invariance or not?

Perform again the LRT for all nested models investigated so far.

```
# LRT.
anova(model_ex_1_configural_fit, model_ex_1_weak_fit, model_ex_1_strong_fit)

## Chi-Squared Difference Test
##
##              Df  AIC   BIC   Chisq Chisq diff Df diff Pr(>Chisq)
## model_ex_1_configural_fit 38 10583 10765  53.380
## model_ex_1_weak_fit      44 10584 10744  65.992    12.613      6 0.04961 *
## model_ex_1_strong_fit    50 10615 10753 109.088    43.096      6 1.117e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We already see that we do not have support for strong measurement invariance. Hence, we should not continue with strict measurement invariance. However, we can identify which intercepts show misfit and allow those intercepts to be freely estimated in both groups. Doing so, we investigate partial strong measurement invariance.

- i. Investigate which intercepts show misfit in the model fit at point (*f*). Free the corresponding intercept to be freely estimated in both groups and compare the fit of this model with that fitted at point (*d*).
- Tip: check `lavaan` functions `parTable` and `lavTestScore`.

We check the misfit by looking at the discrepancy between the estimated intercepts by `lavaan` and the observed means for each group. Then, we can test if our fit would improve with those parameters being freely estimated.

To inspect the estimated intercepts, we use the `lavInspect` function in `lavaan`.

```
# Means discrepancies for group one.
lavInspect(model_ex_1_strong_fit, "mu")[[1]] - group_1_means

##   inf   sim   voc   com p_com p_arr b_des o_ass
## 0.387 -0.955 0.088 0.335 -0.151 -0.504 0.166 0.250
```

```
# Means discrepancies for group two.
lavInspect(model_ex_1_strong_fit, "mu")[[2]] - group_2_means
```

```
##   inf    sim    voc    com p_com p_arr b_des o_ass
## -0.149 0.228 -0.046 -0.256 0.120 0.164 -0.063 -0.113
```

We can see that the misfit is most prominent for variable `sim`. We can now check the improvement in model fit if we freely estimate the intercepts for `sim` across groups (i.e., if we release them from the equality constraint). To do this, we use the `lavTestScore` function in `lavaan`.

First, let us use the `parTable` function in `lavaan` to print the parameter table and learn the label `lavaan` automatically generated for the intercept of variable `sim`. Note that, behind the scenes, `lavaan` applies labels automatically to enforce the equality constraints requested via the argument `group.equal`.

```
# Show the parameter table.
```

```
parTable(model_ex_1_strong_fit)
```

```
##   id   lhs op   rhs user block group free ustart exo label plabel  start  est  se
## 1  1 verbal =~   inf    1    1    1  0    1  0      .p1.  1.000 1.000 0.000
## 2  2 verbal =~   sim    1    1    1  1    NA  0      .p2.  1.340 1.118 0.077
## 3  3 verbal =~   voc    1    1    1  2    NA  0      .p3.  1.148 1.110 0.073
## 4  4 verbal =~   com    1    1    1  3    NA  0      .p4.  0.988 0.858 0.070
## 5  5 visual =~ p_com    1    1    1  0    1  0      .p5.  1.000 1.000 0.000
## 6  6 visual =~ p_arr    1    1    1  4    NA  0      .p6.  1.592 0.892 0.116
## 7  7 visual =~ b_des    1    1    1  5    NA  0      .p7.  1.432 1.188 0.117
## 8  8 visual =~ o_ass    1    1    1  6    NA  0      .p8.  1.357 1.140 0.112
## 9  9 verbal =~ visual  1    1    1  7    NA  0      .p9.  0.000 3.894 0.847
## 10 10 inf  ~~   inf    0    1    1  8    NA  0      .p10. 4.624 4.007 0.743
## 11 11 sim  ~~   sim    0    1    1  9    NA  0      .p11. 6.154 4.146 0.800
## 12 12 voc  ~~   voc    0    1    1 10    NA  0      .p12. 4.994 1.936 0.501
## 13 13 com  ~~   com    0    1    1 11    NA  0      .p13. 4.011 2.595 0.496
## 14 14 p_com ~~ p_com    0    1    1 12    NA  0      .p14. 3.062 2.823 0.605
## 15 15 p_arr ~~ p_arr    0    1    1 13    NA  0      .p15. 7.705 10.327 1.716
## 16 16 b_des ~~ b_des    0    1    1 14    NA  0      .p16. 5.542 5.677 1.091
## 17 17 o_ass ~~ o_ass    0    1    1 15    NA  0      .p17. 4.838 4.570 0.909
## 18 18 verbal ~~ verbal  0    1    1 16    NA  0      .p18. 0.050 6.682 1.333
## 19 19 visual ~~ visual  0    1    1 17    NA  0      .p19. 0.050 3.923 0.893
## 20 20 inf ~1      0    1    1 18    NA  0      .p20. 10.090 10.477 0.321
## 21 21 sim ~1      0    1    1 19    NA  0      .p21. 12.070 11.115 0.351
## 22 22 voc ~1      0    1    1 20    NA  0      .p22. 10.250 10.338 0.341
## 23 23 com ~1      0    1    1 21    NA  0      .p23.  9.960 10.295 0.279
## 24 24 p_com ~1      0    1    1 22    NA  0      .p24. 10.900 10.749 0.265
## 25 25 p_arr ~1      0    1    1 23    NA  0      .p25. 11.240 10.736 0.283
## 26 26 b_des ~1      0    1    1 24    NA  0      .p26. 10.300 10.466 0.321
## 27 27 o_ass ~1      0    1    1 25    NA  0      .p27. 10.440 10.690 0.305
## 28 28 verbal ~1      0    1    1  0    0  0      .p28.  0.000  0.000  0.000
## 29 29 visual ~1      0    1    1  0    0  0      .p29.  0.000  0.000  0.000
## 30 30 verbal =~   inf    1    2    2  0    1  0      .p30.  1.000  1.000  0.000
## 31 31 verbal =~   sim    1    2    2 26    NA  0      .p2.  0.994  1.118  0.077
## 32 32 verbal =~   voc    1    2    2 27    NA  0      .p3.  1.088  1.110  0.073
## 33 33 verbal =~   com    1    2    2 28    NA  0      .p4.  0.783  0.858  0.070
## 34 34 visual =~ p_com    1    2    2  0    1  0      .p34.  1.000  1.000  0.000
## 35 35 visual =~ p_arr    1    2    2 29    NA  0      .p6.  0.705  0.892  0.116
## 36 36 visual =~ b_des    1    2    2 30    NA  0      .p7.  1.213  1.188  0.117
## 37 37 visual =~ o_ass    1    2    2 31    NA  0      .p8.  1.172  1.140  0.112
## 38 38 verbal =~ visual  1    2    2 32    NA  0      .p38.  0.000  3.639  0.562
## 39 39 inf  ~~   inf    0    2    2 33    NA  0      .p39.  4.781  3.799  0.455
## 40 40 sim  ~~   sim    0    2    2 34    NA  0      .p40.  4.184  2.440  0.364
## 41 41 voc  ~~   voc    0    2    2 35    NA  0      .p41.  4.478  2.489  0.364
```

```
## 42 42 com ~~ com 0 2 2 36 NA 0 .p42. 4.184 4.896 0.539
## 43 43 p_com ~~ p_com 0 2 2 37 NA 0 .p43. 5.094 5.534 0.663
## 44 44 p_arr ~~ p_arr 0 2 2 38 NA 0 .p44. 5.418 8.301 0.911
## 45 45 b_des ~~ b_des 0 2 2 39 NA 0 .p45. 5.751 5.293 0.706
## 46 46 o_ass ~~ o_ass 0 2 2 40 NA 0 .p46. 5.418 5.128 0.672
## 47 47 verbal ~~ verbal 0 2 2 41 NA 0 .p47. 0.050 5.133 0.766
## 48 48 visual ~~ visual 0 2 2 42 NA 0 .p48. 0.050 4.321 0.791
## 49 49 inf ~1 0 2 2 43 NA 0 .p20. .p49. 10.100 10.477 0.321
## 50 50 sim ~1 0 2 2 44 NA 0 .p21. .p50. 10.300 11.115 0.351
## 51 51 voc ~1 0 2 2 45 NA 0 .p22. .p51. 9.800 10.338 0.341
## 52 52 com ~1 0 2 2 46 NA 0 .p23. .p52. 10.100 10.295 0.279
## 53 53 p_com ~1 0 2 2 47 NA 0 .p24. .p53. 10.100 10.749 0.265
## 54 54 p_arr ~1 0 2 2 48 NA 0 .p25. .p54. 10.100 10.736 0.283
## 55 55 b_des ~1 0 2 2 49 NA 0 .p26. .p55. 9.900 10.466 0.321
## 56 56 o_ass ~1 0 2 2 50 NA 0 .p27. .p56. 10.200 10.690 0.305
## 57 57 verbal ~1 0 2 2 51 NA 0 .p57. 0.000 -0.525 0.348
## 58 58 visual ~1 0 2 2 52 NA 0 .p58. 0.000 -0.529 0.301
## 59 59 .p2. == .p31. 2 0 0 0 NA 0 0.000 0.000 0.000
## 60 60 .p3. == .p32. 2 0 0 0 NA 0 0.000 0.000 0.000
## 61 61 .p4. == .p33. 2 0 0 0 NA 0 0.000 0.000 0.000
## 62 62 .p6. == .p35. 2 0 0 0 NA 0 0.000 0.000 0.000
## 63 63 .p7. == .p36. 2 0 0 0 NA 0 0.000 0.000 0.000
## 64 64 .p8. == .p37. 2 0 0 0 NA 0 0.000 0.000 0.000
## 65 65 .p20. == .p49. 2 0 0 0 NA 0 0.000 0.000 0.000
## 66 66 .p21. == .p50. 2 0 0 0 NA 0 0.000 0.000 0.000
## 67 67 .p22. == .p51. 2 0 0 0 NA 0 0.000 0.000 0.000
## 68 68 .p23. == .p52. 2 0 0 0 NA 0 0.000 0.000 0.000
## 69 69 .p24. == .p53. 2 0 0 0 NA 0 0.000 0.000 0.000
## 70 70 .p25. == .p54. 2 0 0 0 NA 0 0.000 0.000 0.000
## 71 71 .p26. == .p55. 2 0 0 0 NA 0 0.000 0.000 0.000
## 72 72 .p27. == .p56. 2 0 0 0 NA 0 0.000 0.000 0.000
```

To find the parameter labels, we use check the `plabel` column for the specific relationship (i.e., `lavaan` equation syntax) that we are interested in. We see that for the first group, `lavaan` attached the label `.p21.` for the parameter `sim ~ 1` (i.e., the intercept).

```
parTable(model_ex_1_strong_fit)[21, ]
```

```
## id lhs op rhs user block group free ustart exo label plabel start est se
## 21 21 sim ~1 0 1 1 19 NA 0 .p21. .p21. 12.07 11.115 0.351
```

In the second group, `lavaan` refers to the parameter `sim ~ 1` (i.e., the intercept) using the label `.p50.`

```
parTable(model_ex_1_strong_fit)[50, ]
```

```
## id lhs op rhs user block group free ustart exo label plabel start est se
## 50 50 sim ~1 0 2 2 44 NA 0 .p21. .p50. 10.3 11.115 0.351
```

In order for `lavaan` to respect the equality constraints we requested, it set both of these labels to be equal, i.e., `.p21. == .p50.` Therefore, when looking at the output of `lavTestScore` to understand which “release of constraints” results in the highest change in the χ^2 difference, we expect `.p21. == .p50.` to yield the highest change.

```
# Perform the score test for releasing constrained model parameters.
lavTestScore(model_ex_1_strong_fit)
```

```
## $test
##
## total score test:
##
```

```
##      test      X2 df p.value
## 1 score 50.655 14      0
##
## $uni
##
## univariate score tests:
##
##      lhs op   rhs      X2 df p.value
## 1 .p2. == .p31.  0.634  1  0.426
## 2 .p3. == .p32.  0.138  1  0.710
## 3 .p4. == .p33.  1.649  1  0.199
## 4 .p6. == .p35.  5.483  1  0.019
## 5 .p7. == .p36.  0.017  1  0.898
## 6 .p8. == .p37.  0.012  1  0.913
## 7 .p20. == .p49.  5.165  1  0.023
## 8 .p21. == .p50. 29.459  1  0.000
## 9 .p22. == .p51.  0.850  1  0.357
## 10 .p23. == .p52.  7.455  1  0.006
## 11 .p24. == .p53.  1.745  1  0.186
## 12 .p25. == .p54.  2.955  1  0.086
## 13 .p26. == .p55.  0.772  1  0.380
## 14 .p27. == .p56.  2.376  1  0.123
```

The overall `total score test` is a multivariate test indicating whether releasing (i.e., freeing) all constraints improves the fit over the base model. We reject the null hypothesis and proceed by looking at the univariate tests. Here, we indeed see that freeing the constraint `.p21. == .p50.` would result in the largest χ^2 difference (e.g., 29.459). We go ahead and refit the strong measurement invariance model with the `sim ~ 1` freely estimated in both groups. Then, we check again using the function `lavTestScore` whether additional constraints can be freed. To do so, we use the `group.partial` argument. Note that, in this case, we are actually investigating what is called *strong partial measurement invariance*.

```
# Fit the model.
model_ex_1_strong_partial_fit <- sem(
  model_ex_1,
  sample.cov = covariances,
  sample.nobs = samples,
  sample.mean = means,
  group.equal = c("loadings", "intercepts"),
  group.partial = c("sim ~ 1")
)

# Fit measures.
fitMeasures(model_ex_1_strong_partial_fit, fit.measures = fit_indices)
```

```
##      chisq      df      pvalue      cfi      tli      rmsea rmsea.pvalue
##      75.924     49.000      0.008     0.974     0.970     0.063      0.218
##      srmr
##      0.058
```

```
# Perform the score test.
lavTestScore(model_ex_1_strong_partial_fit)
```

```
## $test
##
## total score test:
##
##      test      X2 df p.value
## 1 score 22.189 13  0.053
##
```

```
## $uni
##
## univariate score tests:
##
##      lhs op   rhs      X2 df p.value
## 1 .p2. == .p31. 2.561  1  0.110
## 2 .p3. == .p32. 0.669  1  0.413
## 3 .p4. == .p33. 0.952  1  0.329
## 4 .p6. == .p35. 5.477  1  0.019
## 5 .p7. == .p36. 0.016  1  0.899
## 6 .p8. == .p37. 0.010  1  0.922
## 7 .p20. == .p49. 0.678  1  0.410
## 8 .p22. == .p51. 3.610  1  0.057
## 9 .p23. == .p52. 1.967  1  0.161
## 10 .p24. == .p53. 1.741  1  0.187
## 11 .p25. == .p54. 2.951  1  0.086
## 12 .p26. == .p55. 0.778  1  0.378
## 13 .p27. == .p56. 2.366  1  0.124
```

This time we see that the multivariate score test is not significant and looking at the univariate tests we cannot free any parameters. We perform again the *LRT* for the nested models investigated so far.

```
# LRT.
anova(model_ex_1_configural_fit, model_ex_1_weak_fit, model_ex_1_strong_partial_fit)
```

```
## Chi-Squared Difference Test
##
##              Df    AIC    BIC  Chisq Chisq diff Df diff Pr(>Chisq)
## model_ex_1_configural_fit    38 10583 10765 53.380
## model_ex_1_weak_fit          44 10584 10744 65.992    12.6128      6  0.04961 *
## model_ex_1_strong_partial_fit 49 10584 10726 75.924     9.9317      5  0.07720 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Based on the χ^2 difference test above and the fit measures below we see some support for partial scalar measurement invariance.

```
# Combine all fit measures for all models into a data frame.
fit_measures_all_ex_1 <- rbind(
  configural = fitMeasures(model_ex_1_configural_fit, fit_indices),
  weak = fitMeasures(model_ex_1_weak_fit, fit_indices),
  strong = fitMeasures(model_ex_1_strong_fit, fit_indices),
  strong_partial = fitMeasures(model_ex_1_strong_partial_fit, fit_indices)
)

# Round fit measures for all models to four decimals.
print(round(fit_measures_all_ex_1, 3))
```

```
##              chisq df pvalue   cfi   tli rmsea rmsea.pvalue srmr
## configural    53.380 38  0.050 0.985 0.978 0.054      0.402 0.034
## weak          65.992 44  0.018 0.979 0.973 0.060      0.281 0.055
## strong        109.088 50  0.000 0.942 0.935 0.092      0.003 0.064
## strong_partial 75.924 49  0.008 0.974 0.970 0.063      0.218 0.058
```

Or print the fit measures as an APA-style table.

```
# Print pretty table.
knitr::kable(round(fit_measures_all_ex_1, 3), caption = "Fit measures for all models investigated.")
```


Table 1: Fit measures for all models investigated.

	chisq	df	pvalue	cfi	tli	rmsea	rmsea.pvalue	srmr
configural	53.380	38	0.050	0.985	0.978	0.054	0.402	0.034
weak	65.992	44	0.018	0.979	0.973	0.060	0.281	0.055
strong	109.088	50	0.000	0.942	0.935	0.092	0.003	0.064
strong_partial	75.924	49	0.008	0.974	0.970	0.063	0.218	0.058

j. What kind of comparisons can we make when we find support for strong (partial) measurement invariance?

When we find support for strong measurement invariance, we can compare the means of the latent variables across groups. For our example, we see that children in the second group (i.e., the norming group) perform worse than children in the first group (i.e., with manic symptoms) on both the verbal (i.e., -0.171) and visual (i.e., -0.53) components of intelligence. However, if we look at the p -values we see that neither of these differences are significant (e.g., $p = 0.633$ for verbal and $p = 0.078$ for visual).

Additional information on the p -values:

```
# Get the parameters.
par <- parameterTable(model_ex_1_strong_partial_fit)

# Filter the parameters for ease of read.
# Group one should be reference, hence zero.
par[par$op == "-1" & par$group == 1 & (par$lhs == "verbal" | par$lhs == "visual"), ]

##   id   lhs op rhs user block group free ustart exo label plabel start est se
## 28 28 verbal -1     0    1    1    0    0    0    .p28.    0    0    0
## 29 29 visual -1     0    1    1    0    0    0    .p29.    0    0    0

# Group two should show be relative to group one.
par[par$op == "-1" & par$group == 2 & (par$lhs == "verbal" | par$lhs == "visual"), ]

##   id   lhs op rhs user block group free ustart exo label plabel start est se
## 57 57 verbal -1     0    2    2   51    NA    0    .p57.    0 -0.171 0.358
## 58 58 visual -1     0    2    2   52    NA    0    .p58.    0 -0.530 0.301

# Or, we know that the intercepts are named `alpha` in SEM, hence we can look for
# that in the output of `inspect()`.
inspect(model_ex_1_strong_partial_fit, what = "est")

## $group_1
## $group_1$lambda
##      verbal visual
## inf      1.000 0.000
## sim      1.107 0.000
## voc      1.096 0.000
## com      0.862 0.000
## p_com    0.000 1.000
## p_arr    0.000 0.891
## b_des    0.000 1.188
## o_ass    0.000 1.139
##
## $group_1$theta
##      inf      sim      voc      com      p_com      p_arr      b_des      o_ass
```

```

## inf      3.750
## sim      0.000  2.768
## voc      0.000  0.000  2.160
## com      0.000  0.000  0.000  2.455
## p_com    0.000  0.000  0.000  0.000  2.814
## p_arr    0.000  0.000  0.000  0.000  0.000  10.333
## b_des    0.000  0.000  0.000  0.000  0.000  0.000  5.675
## o_ass    0.000  0.000  0.000  0.000  0.000  0.000  0.000  4.588
##
## $group_1$psi
##      verbal visual
## verbal 6.908
## visual 3.929  3.925
##
## $group_1$nu
##      intrcp
## inf      10.219
## sim      12.070
## voc      10.072
## com      10.120
## p_com    10.750
## p_arr    10.736
## b_des    10.466
## o_ass    10.690
##
## $group_1$alpha
##      intrcp
## verbal      0
## visual      0
##
##
## $group_2
## $group_2$lambda
##      verbal visual
## inf      1.000  0.000
## sim      1.107  0.000
## voc      1.096  0.000
## com      0.862  0.000
## p_com    0.000  1.000
## p_arr    0.000  0.891
## b_des    0.000  1.188
## o_ass    0.000  1.139
##
## $group_2$theta
##      inf  sim  voc  com  p_com p_arr b_des o_ass
## inf      3.768
## sim      0.000  2.343
## voc      0.000  0.000  2.528
## com      0.000  0.000  0.000  4.841
## p_com    0.000  0.000  0.000  0.000  5.533
## p_arr    0.000  0.000  0.000  0.000  0.000  8.302
## b_des    0.000  0.000  0.000  0.000  0.000  0.000  5.288
## o_ass    0.000  0.000  0.000  0.000  0.000  0.000  0.000  5.133
##
## $group_2$psi
##      verbal visual
## verbal 5.212
## visual 3.671  4.326
##
## $group_2$nu
##      intrcp

```

```
## inf 10.219
## sim 10.489
## voc 10.072
## com 10.120
## p_com 10.750
## p_arr 10.736
## b_des 10.466
## o_ass 10.690
##
## $group_2$alpha
##      intrcp
## verbal -0.171
## visual -0.530

# $group_2$alpha
#      intrcp
# verbal -0.171
# visual -0.530

# Or, if you want the standard errors, replace `what = "est"` with `what = "se`.
inspect(model_ex_1_strong_partial_fit, what = "se")
```

```
## $group_1
## $group_1$lambda
##      verbal visual
## inf 0.000 0.000
## sim 0.073 0.000
## voc 0.072 0.000
## com 0.068 0.000
## p_com 0.000 0.000
## p_arr 0.000 0.115
## b_des 0.000 0.117
## o_ass 0.000 0.111
##
## $group_1$theta
##      inf sim voc com p_com p_arr b_des o_ass
## inf 0.697
## sim 0.000 0.589
## voc 0.000 0.000 0.505
## com 0.000 0.000 0.000 0.469
## p_com 0.000 0.000 0.000 0.000 0.604
## p_arr 0.000 0.000 0.000 0.000 0.000 1.717
## b_des 0.000 0.000 0.000 0.000 0.000 0.000 1.091
## o_ass 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.912
##
## $group_1$psi
##      verbal visual
## verbal 1.361
## visual 0.856 0.893
##
## $group_1$nu
##      intrcp
## inf 0.327
## sim 0.372
## voc 0.347
## com 0.284
## p_com 0.265
## p_arr 0.283
## b_des 0.321
## o_ass 0.305
##
```

```

## $group_1$alpha
##      intrcp
## verbal    0
## visual    0
##
##
## $group_2
## $group_2$lambda
##      verbal visual
## inf      0.000 0.000
## sim      0.073 0.000
## voc       0.072 0.000
## com       0.068 0.000
## p_com     0.000 0.000
## p_arr     0.000 0.115
## b_des     0.000 0.117
## o_ass     0.000 0.111
##
## $group_2$theta
##      inf  sim  voc  com  p_com p_arr b_des o_ass
## inf    0.452
## sim    0.000 0.352
## voc    0.000 0.000 0.363
## com    0.000 0.000 0.000 0.534
## p_com  0.000 0.000 0.000 0.000 0.663
## p_arr  0.000 0.000 0.000 0.000 0.000 0.910
## b_des  0.000 0.000 0.000 0.000 0.000 0.000 0.706
## o_ass  0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.672
##
## $group_2$psi
##      verbal visual
## verbal 0.769
## visual 0.565 0.792
##
## $group_2$nu
##      intrcp
## inf      0.327
## sim      0.370
## voc      0.347
## com      0.284
## p_com    0.265
## p_arr    0.283
## b_des    0.321
## o_ass    0.305
##
## $group_2$alpha
##      intrcp
## verbal 0.358
## visual 0.301
# $group_2$alpha
#      intrcp
# verbal 0.358
# visual 0.301

# You can find the standard errors p-values in the summary output under the `Intercepts` section.
summary(model_ex_1_strong_partial_fit)

```

```

## lavaan 0.6-12 ended normally after 112 iterations
##
##      Estimator                                ML

```

```

## Optimization method NLMINB
## Number of model parameters 52
## Number of equality constraints 13
##
## Number of observations per group:
## group_1 81
## group_2 200
##
## Model Test User Model:
##
## Test statistic 75.924
## Degrees of freedom 49
## P-value (Chi-square) 0.008
## Test statistic for each group:
## group_1 44.340
## group_2 31.584
##
## Parameter Estimates:
##
## Standard errors Standard
## Information Expected
## Information saturated (h1) model Structured
##
##
## Group 1 [group_1]:
##
## Latent Variables:
## Estimate Std.Err z-value P(>|z|)
## verbal =~
## inf 1.000
## sim (.p2.) 1.107 0.073 15.181 0.000
## voc (.p3.) 1.096 0.072 15.253 0.000
## com (.p4.) 0.862 0.068 12.658 0.000
## visual =~
## p_com 1.000
## p_arr (.p6.) 0.891 0.115 7.722 0.000
## b_des (.p7.) 1.188 0.117 10.193 0.000
## o_ass (.p8.) 1.139 0.111 10.219 0.000
##
## Covariances:
## Estimate Std.Err z-value P(>|z|)
## verbal ~~
## visual 3.929 0.856 4.592 0.000
##
## Intercepts:
## Estimate Std.Err z-value P(>|z|)
## .inf (.20.) 10.219 0.327 31.211 0.000
## .sim 12.070 0.372 32.403 0.000
## .voc (.22.) 10.072 0.347 29.041 0.000
## .com (.23.) 10.120 0.284 35.622 0.000
## .p_com (.24.) 10.750 0.265 40.570 0.000
## .p_arr (.25.) 10.736 0.283 37.950 0.000
## .b_des (.26.) 10.466 0.321 32.647 0.000
## .o_ass (.27.) 10.690 0.305 35.049 0.000
## verbal 0.000
## visual 0.000
##
## Variances:
## Estimate Std.Err z-value P(>|z|)
## .inf 3.750 0.697 5.381 0.000
## .sim 2.768 0.589 4.704 0.000

```

```

##      .voc      2.160    0.505    4.279    0.000
##      .com      2.455    0.469    5.231    0.000
##      .p_com    2.814    0.604    4.655    0.000
##      .p_arr    10.333    1.717    6.017    0.000
##      .b_des     5.675    1.091    5.200    0.000
##      .o_ass     4.588    0.912    5.029    0.000
##      verbal     6.908    1.361    5.076    0.000
##      visual     3.925    0.893    4.393    0.000
##
##
## Group 2 [group_2]:
##
## Latent Variables:
##      Estimate Std.Err z-value P(>|z|)
##      verbal =~
##      inf      1.000
##      sim      (.p2.) 1.107    0.073    15.181    0.000
##      voc      (.p3.) 1.096    0.072    15.253    0.000
##      com      (.p4.) 0.862    0.068    12.658    0.000
##      visual =~
##      p_com     1.000
##      p_arr     (.p6.) 0.891    0.115    7.722    0.000
##      b_des     (.p7.) 1.188    0.117    10.193    0.000
##      o_ass     (.p8.) 1.139    0.111    10.219    0.000
##
## Covariances:
##      Estimate Std.Err z-value P(>|z|)
##      verbal ~~
##      visual     3.671    0.565    6.499    0.000
##
## Intercepts:
##      Estimate Std.Err z-value P(>|z|)
##      .inf      (.20.) 10.219    0.327    31.211    0.000
##      .sim      10.489    0.370    28.342    0.000
##      .voc      (.22.) 10.072    0.347    29.041    0.000
##      .com      (.23.) 10.120    0.284    35.622    0.000
##      .p_com    (.24.) 10.750    0.265    40.570    0.000
##      .p_arr    (.25.) 10.736    0.283    37.950    0.000
##      .b_des    (.26.) 10.466    0.321    32.647    0.000
##      .o_ass    (.27.) 10.690    0.305    35.049    0.000
##      verbal     -0.171    0.358    -0.477    0.633
##      visual     -0.530    0.301    -1.761    0.078
##
## Variances:
##      Estimate Std.Err z-value P(>|z|)
##      .inf      3.768    0.452    8.334    0.000
##      .sim      2.343    0.352    6.646    0.000
##      .voc      2.528    0.363    6.958    0.000
##      .com      4.841    0.534    9.065    0.000
##      .p_com    5.533    0.663    8.346    0.000
##      .p_arr    8.302    0.910    9.119    0.000
##      .b_des    5.288    0.706    7.494    0.000
##      .o_ass    5.133    0.672    7.637    0.000
##      verbal     5.212    0.769    6.776    0.000
##      visual     4.326    0.792    5.461    0.000

```

```

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

```

```
#      verbal      -0.171    0.358   -0.477    0.633
#      visual      -0.530    0.301   -1.761    0.078

# Or, you can approximate the p-values yourself given that you have the standard errors.
# For `verbal` latent variable.
pnorm(-0.171 / 0.358) * 2
```

```
## [1] 0.6328968
```

```
# [1] 0.6328968
```

```
# For `visual` latent variable.
```

```
pnorm(-0.530 / 0.301) * 2
```

```
## [1] 0.07827271
```

```
# [1] 0.07827271
```

- k. Investigate *strict* measurement invariance. Also, perform a *LRT* to compare the strict measurement invariance model to the strong partial measurement invariance model. Report and interpret model fit of the fitted model and the results of the *LRT*.

To investigate strict measurement invariance, on top of constraining the loadings and the intercepts to be equal in both groups, we also need to constrain the residual covariances. To achieve this, we can, again, use the `group.equal` argument in `lavaan`.

```
# Fit the model.
model_ex_1_strict_fit <- sem(
  model_ex_1,
  sample.cov = covariances,
  sample.nobs = samples,
  sample.mean = means,
  group.equal = c("loadings", "intercepts", "residuals"),
  group.partial = c("sim ~ 1")
)

# Model summary.
summary(model_ex_1_strict_fit, standardized = TRUE)
```

```
## lavaan 0.6-12 ended normally after 79 iterations
##
##      Estimator                      ML
##      Optimization method          NLMINB
##      Number of model parameters          52
##      Number of equality constraints        21
##
##      Number of observations per group:
##      group_1                          81
##      group_2                         200
##
## Model Test User Model:
##
##      Test statistic                  94.881
##      Degrees of freedom                57
##      P-value (Chi-square)              0.001
```

```

## Test statistic for each group:
##   group_1                      59.688
##   group_2                      35.193
##
## Parameter Estimates:
##
##   Standard errors                      Standard
##   Information                      Expected
##   Information saturated (h1) model      Structured
##
##
## Group 1 [group_1]:
##
## Latent Variables:
##           Estimate Std.Err z-value P(>|z|) Std.lv Std.all
##   verbal =~
##     inf           1.000
##     sim   (.p2.)   1.109   0.073  15.242   0.000   2.912   0.880
##     voc    (.p3.)   1.096   0.072  15.245   0.000   2.878   0.880
##     com    (.p4.)   0.846   0.070  12.098   0.000   2.222   0.736
##   visual =~
##     p_com           1.000
##     p_arr   (.p6.)   0.893   0.115   7.786   0.000   1.809   0.518
##     b_des   (.p7.)   1.179   0.115  10.244   0.000   2.389   0.720
##     o_ass   (.p8.)   1.128   0.110  10.218   0.000   2.284   0.718
##
## Covariances:
##           Estimate Std.Err z-value P(>|z|) Std.lv Std.all
##   verbal ~~
##     visual           3.936   0.886   4.443   0.000   0.740   0.740
##
## Intercepts:
##           Estimate Std.Err z-value P(>|z|) Std.lv Std.all
##   .inf   (.20.)   10.229   0.328  31.139   0.000  10.229   3.136
##   .sim           12.070   0.368  32.834   0.000  12.070   3.648
##   .voc    (.22.)   10.074   0.349  28.828   0.000  10.074   3.081
##   .com    (.23.)   10.171   0.287  35.406   0.000  10.171   3.370
##   .p_com   (.24.)   10.700   0.281  38.130   0.000  10.700   3.598
##   .p_arr   (.25.)   10.759   0.286  37.633   0.000  10.759   3.083
##   .b_des   (.26.)   10.451   0.324  32.302   0.000  10.451   3.152
##   .o_ass   (.27.)   10.686   0.310  34.487   0.000  10.686   3.358
##   verbal           0.000
##   visual           0.000
##
## Variances:
##           Estimate Std.Err z-value P(>|z|) Std.lv Std.all
##   .inf   (.10.)   3.737   0.382   9.778   0.000   3.737   0.351
##   .sim   (.11.)   2.465   0.312   7.906   0.000   2.465   0.225
##   .voc   (.12.)   2.411   0.305   7.912   0.000   2.411   0.225
##   .com   (.13.)   4.172   0.396  10.542   0.000   4.172   0.458
##   .p_com (.14.)   4.744   0.502   9.442   0.000   4.744   0.536

```



```

##      .p_arr   (.15.)    8.908    0.821   10.848    0.000    8.908    0.731
##      .b_des   (.16.)    5.289    0.599    8.830    0.000    5.289    0.481
##      .o_ass   (.17.)    4.911    0.553    8.877    0.000    4.911    0.485
##      verbal                6.901    1.359    5.077    0.000    1.000    1.000
##      visual                4.103    0.981    4.184    0.000    1.000    1.000
##
##
## Group 2 [group_2]:
##
## Latent Variables:
##      Estimate Std.Err z-value P(>|z|) Std.lv Std.all
##      verbal =~
##      inf                1.000                2.292    0.764
##      sim   (.p2.)    1.109    0.073   15.242    0.000    2.540    0.851
##      voc   (.p3.)    1.096    0.072   15.245    0.000    2.511    0.850
##      com   (.p4.)    0.846    0.070   12.098    0.000    1.938    0.688
##      visual =~
##      p_com                1.000                2.109    0.696
##      p_arr   (.p6.)    0.893    0.115    7.786    0.000    1.884    0.534
##      b_des   (.p7.)    1.179    0.115   10.244    0.000    2.487    0.734
##      o_ass   (.p8.)    1.128    0.110   10.218    0.000    2.379    0.732
##
## Covariances:
##      Estimate Std.Err z-value P(>|z|) Std.lv Std.all
##      verbal ~~
##      visual                3.712    0.567    6.546    0.000    0.768    0.768
##
## Intercepts:
##      Estimate Std.Err z-value P(>|z|) Std.lv Std.all
##      .inf   (.20.)   10.229    0.328   31.139    0.000   10.229    3.412
##      .sim                10.505    0.374   28.088    0.000   10.505    3.518
##      .voc   (.22.)   10.074    0.349   28.828    0.000   10.074    3.412
##      .com   (.23.)   10.171    0.287   35.406    0.000   10.171    3.612
##      .p_com (.24.)   10.700    0.281   38.130    0.000   10.700    3.529
##      .p_arr (.25.)   10.759    0.286   37.633    0.000   10.759    3.048
##      .b_des (.26.)   10.451    0.324   32.302    0.000   10.451    3.085
##      .o_ass (.27.)   10.686    0.310   34.487    0.000   10.686    3.287
##      verbal                -0.185    0.361   -0.513    0.608   -0.081   -0.081
##      visual                -0.520    0.308   -1.685    0.092   -0.246   -0.246
##
## Variances:
##      Estimate Std.Err z-value P(>|z|) Std.lv Std.all
##      .inf   (.10.)    3.737    0.382    9.778    0.000    3.737    0.416
##      .sim   (.11.)    2.465    0.312    7.906    0.000    2.465    0.276
##      .voc   (.12.)    2.411    0.305    7.912    0.000    2.411    0.277
##      .com   (.13.)    4.172    0.396   10.542    0.000    4.172    0.526
##      .p_com (.14.)    4.744    0.502    9.442    0.000    4.744    0.516
##      .p_arr (.15.)    8.908    0.821   10.848    0.000    8.908    0.715
##      .b_des (.16.)    5.289    0.599    8.830    0.000    5.289    0.461
##      .o_ass (.17.)    4.911    0.553    8.877    0.000    4.911    0.465
##      verbal                5.252    0.774    6.789    0.000    1.000    1.000

```

```
##      visual      4.449    0.794    5.606    0.000    1.000    1.000
```

```
# Fit measures.
```

```
fitMeasures(model_ex_1_strict_fit, fit.measures = fit_indices)
```

```
##      chisq      df      pvalue      cfi      tli      rmsea rmsea.pvalue
```

```
##      94.881    57.000    0.001    0.963    0.964    0.069    0.105
```

```
##      srmr
```

```
##      0.070
```

```
# Save the model plots for each groups as a list with two elements.
```

```
plots_strict <- semPaths(
```

```
  model_ex_1_strict_fit,
```

```
  what = "paths",
```

```
  whatLabels = "est",
```

```
  DoNotPlot = TRUE,
```

```
  ask = FALSE,
```

```
  title = FALSE
```

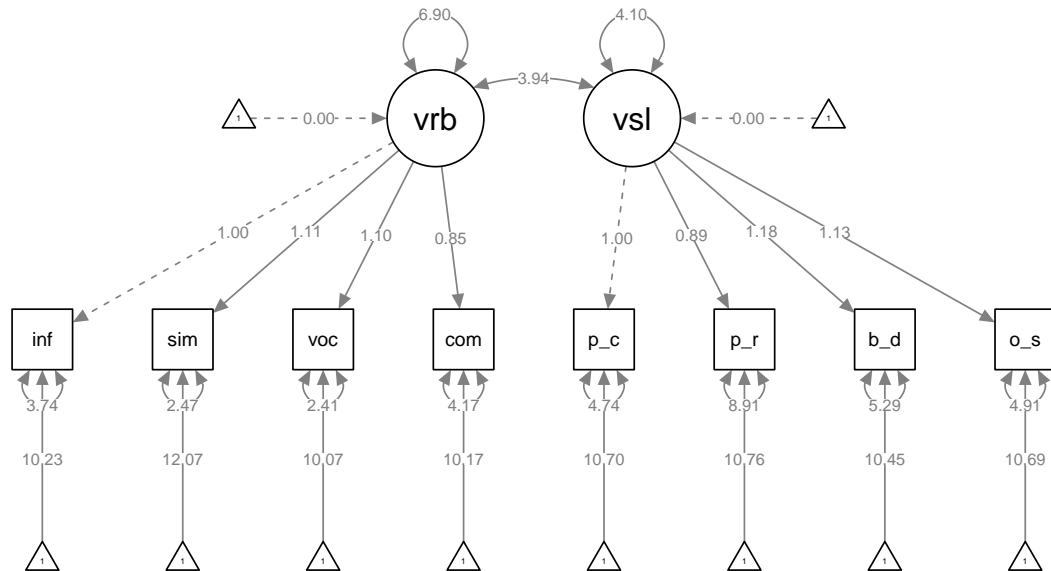
```
)
```

```
# Plot the model for the first group.
```

```
plot(plots_strict[[1]])
```

```
title("Strict MI | Group 1", adj = 0)
```

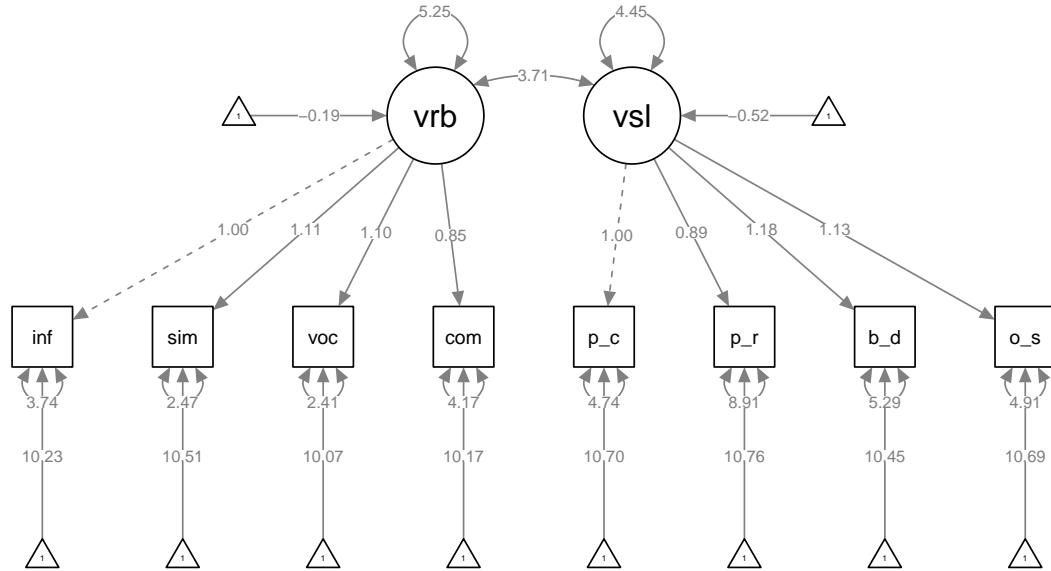
Strict MI | Group 1



```
# Plot the model for the second group.
```

```
plot(plots_strict[[2]])
```

```
title("Strict MI | Group 2", adj = 0)
```



We can again combine the fit measures for the models investigated so far for convenience.

```
# Combine all fit measures.
fit_measures_all_ex_1 <- rbind(
  fit_measures_all_ex_1,
  strict = fitMeasures(model_ex_1_strict_fit, fit_indices)
)

# Print the fit measures for all models with three decimals.
print(round(fit_measures_all_ex_1, 3))
```

```
##               chisq df pvalue   cfi   tli rmsea rmsea.pvalue  srmr
## configural    53.380 38  0.050 0.985 0.978 0.054      0.402 0.034
## weak          65.992 44  0.018 0.979 0.973 0.060      0.281 0.055
## strong        109.088 50  0.000 0.942 0.935 0.092      0.003 0.064
## strong_partial 75.924 49  0.008 0.974 0.970 0.063      0.218 0.058
## strict        94.881 57  0.001 0.963 0.964 0.069      0.105 0.070
```

Or, again, show the measures in an APA-style table.

```
# Print pretty table.
knitr::kable(round(fit_measures_all_ex_1, 3), caption = "Fit measures for all models investigated.")
```

Table 2: Fit measures for all models investigated.

	chisq	df	pvalue	cfi	tli	rmsea	rmsea.pvalue	srmr
configural	53.380	38	0.050	0.985	0.978	0.054	0.402	0.034
weak	65.992	44	0.018	0.979	0.973	0.060	0.281	0.055
strong	109.088	50	0.000	0.942	0.935	0.092	0.003	0.064
strong_partial	75.924	49	0.008	0.974	0.970	0.063	0.218	0.058
strict	94.881	57	0.001	0.963	0.964	0.069	0.105	0.070

We observe a $\chi^2 = 94.881$ with $DF = 57$ and a p -value = 0.001. The fit indices indicate some support for strict measurement invariance.

Finally, we perform a *LRT* between the strict model and the partial strong model.

```
# Perform LRT.
anova(
  model_ex_1_configural_fit,
  model_ex_1_weak_fit,
  model_ex_1_strong_partial_fit,
  model_ex_1_strict_fit
)

## Chi-Squared Difference Test
##
##
##          Df   AIC   BIC  Chisq Chisq diff Df diff Pr(>Chisq)
## model_ex_1_configural_fit    38 10583 10765 53.380
## model_ex_1_weak_fit         44 10584 10744 65.992    12.6128      6   0.04961 *
## model_ex_1_strong_partial_fit 49 10584 10726 75.924     9.9317      5   0.07720 .
## model_ex_1_strict_fit        57 10587 10700 94.881    18.9574      8   0.01509 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Based on the χ^2 difference test and the fit measures above, we do not see support for strict measurement invariance. We could, of course, use the same strategy as above (i.e., using the function `lavTestScore`) to detect which parameters to free in order to increase the model fit, but this exercise is already getting too long...

1. What kinds of comparisons does strict measurement invariance allow you to make?

Strict measurement invariance is an important prerequisite if we want to make comparisons based on the raw scores of variables (e.g., sum scores of variables). The rationale behind this is that an observed score X consists of a true score T and an error score E (i.e., $X = T + E$; [see this in the context of reliability](#)). Therefore, with strict measurement invariance we ensure that the error part (i.e., E) is the same across groups, allowing us to validly compare sum scores or other measures derived based the raw variable realizations.

Part 3. Generating some data.

Optional.

In *Part 2* of this exercise we used the mean and covariance vectors and matrices for multiple group *CFA*. In `lavaan`, we can also use the entire data for multiple group *CFA*, in which case we can specify the `group` argument to tell `lavaan` which variable to use to split our dataset. This implies that we have an additional variable in our data which indicates the group membership (e.g., manic vs. norming). In this part you will first learn how to generate a complete dataset using reported means and covariances, then you will see how the generated data can be used to test for configural measurement invariance. The reminder of the measurement invariance checks (i.e., weak, strong and strict measurement invariance) are left for you to do as an exercise. Note that this part is optional and you can safely skip it. It is presented here as an example for those of you who are curious about generating data.

To generate our complete datasets, we will use the means and covariances presented in *Figure 1*. Since we already have the observed means and covariances, and we assume that our data comes from a multivariate normal distribution, we can “draw” a sample from this distribution using the observed means and covariances. Recall that, just like the normal distribution, the multivariate normal distribution depends on two parameters, the μ vector of means and the Σ variance-covariance matrix:

$$p(x; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{n/2} |\boldsymbol{\Sigma}|^{1/2}} \exp \left(-\frac{1}{2} (x - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (x - \boldsymbol{\mu}) \right)$$

Conceptually, this means that we use our observed means and covariances as population parameters and we draw a sample from that population (e.g., imagine applying a questionnaire). If the sample we drew is sufficiently large, then the means and covariances we calculate on the generated data should be close to our observed means and covariances that we used as population parameters. Furthermore, since each group has its own values for the mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$, we consider that each group comes from its own population. Therefore, we draw two samples from the multivariate normal distribution, one per group.

That being said, we can go ahead and generate some data. First, let us check that we can indeed recover the means and covariances provided in *Figure 1* if our sample is sufficiently large. We do this as an example for the first group. As always, make sure to check the documentation for function `?rmvnorm` in the R package `mvtnorm`.

```
# Inspect the means for group one.
group_1_means

##   inf   sim   voc   com p_com p_arr b_des o_ass
## 10.09 12.07 10.25   9.96 10.90 11.24 10.30 10.44

# Inspect the covariances for group one.
group_1_cov

##           inf   sim   voc   com p_com p_arr b_des o_ass
## inf   9.364  7.777  6.422 5.669 3.048  3.505  3.690 3.640
## sim   7.777 12.461  8.756 7.445 4.922  4.880  5.440 4.641
## voc   6.422  8.756 10.112 6.797 4.513  4.899  5.220 4.877
## com   5.669  7.445  6.797 8.123 4.116  5.178  3.151 3.568
## p_com 3.048  4.922  4.513 4.116 6.200  5.114  3.587 3.819
## p_arr 3.505  4.880  4.899 5.178 5.114 15.603  6.219 5.811
## b_des 3.690  5.440  5.220 3.151 3.587  6.219 11.223 6.501
## o_ass 3.640  4.641  4.877 3.568 3.819  5.811  6.501 9.797

# Draw a sample of one million respondents for group one.
data_ex_1_group_1 <- rmvnorm(n = 1e6, mean = group_1_means, sigma = group_1_cov)

# Check that we indeed have one million respondents for eight variables.
dim(data_ex_1_group_1)

## [1] 1000000      8

# Compare the means of the variables in the generated data with the provided means.
round(group_1_means - colMeans(data_ex_1_group_1), 4)

##   inf   sim   voc   com p_com p_arr b_des o_ass
## -0.0006 -0.0010  0.0007  0.0028  0.0006 -0.0018  0.0004  0.0007

# Compare the covariances of the generated data with the provided covariance matrix.
round(group_1_cov - cov(data_ex_1_group_1), 3)

##           inf   sim   voc   com p_com p_arr b_des o_ass
## inf   -0.005 -0.002 -0.002 -0.008 -0.006 -0.012 -0.011 -0.007
## sim   -0.002 -0.006  0.001 -0.003 -0.002 -0.010 -0.015 -0.008
## voc   -0.002  0.001  0.010  0.000 -0.003 -0.001 -0.013 -0.005
## com   -0.008 -0.003  0.000 -0.010 -0.010 -0.022 -0.010 -0.009
## p_com -0.006 -0.002 -0.003 -0.010  0.001 -0.010 -0.009  0.004
## p_arr -0.012 -0.010 -0.001 -0.022 -0.010 -0.023 -0.006  0.004
## b_des -0.011 -0.015 -0.013 -0.010 -0.009 -0.006 -0.034 -0.019
## o_ass -0.007 -0.008 -0.005 -0.009  0.004  0.004 -0.019  0.013
```

We can see that the differences are quite small. You can test that the more we increase the sample size (i.e., n), the smaller the differences become.

Now, let us go ahead and take more reasonably sized samples for both groups and combine them into a single data frame. We will take a sample of $N = 100$ for group one and $N = 300$ for group two. Since we are dealing with random number generation (e.g., see `?RNG`), we also set a “seed” so we can replicate the results.

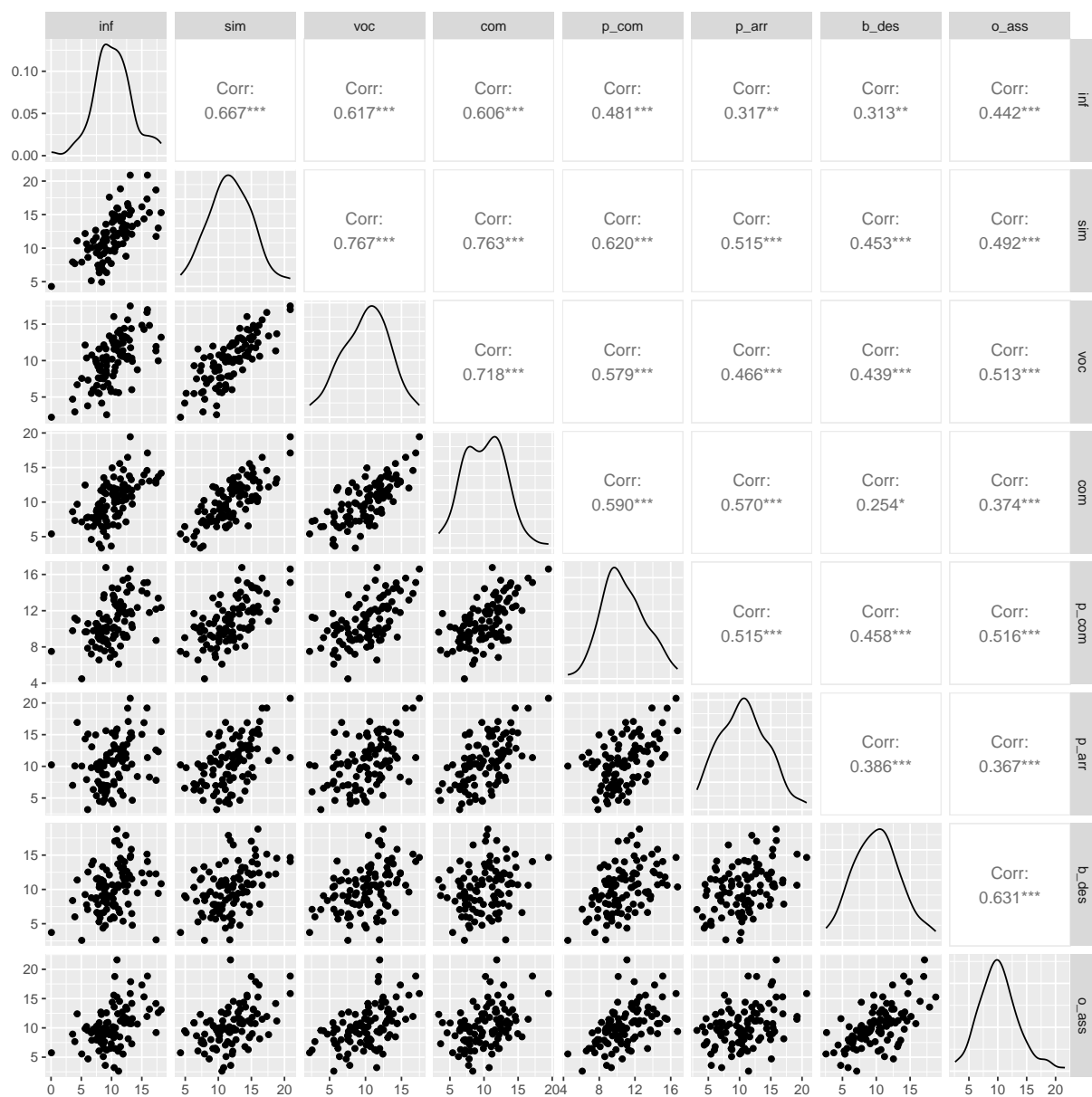
```
# Set seed for the `RNG`.
set.seed(20031993)

# Data for group one with 100 cases.
data_ex_1_group_1 <- rmvnorm(n = 100, mean = group_1_means, sigma = group_1_cov)

# Data for group two with 300 cases.
data_ex_1_group_2 <- rmvnorm(n = 300, mean = group_2_means, sigma = group_2_cov)
```

Before combining the data, we may also want to visualize what we generated. We use the function `ggpairs` in package `GGally`. Note that the function expects the data to be provided as a data frame and not matrix object.

```
# Generated data for group one.
ggpairs(as.data.frame(data_ex_1_group_1))
```



```
# Generated data for group two.
ggpairs(as.data.frame(data_ex_1_group_2))
```



We continue with combining the data for both groups into a single data frame.

```
# Combine the two datasets into a single dataset.
data_ex_1 <- data.frame(rbind(data_ex_1_group_1, data_ex_1_group_2))

# Create a variable that holds the group membership.
group <- as.factor(c(rep("manic", 100), rep("norming", 300)))

# Inspect the `group` variable.
str(group)

## Factor w/ 2 levels "manic","norming": 1 1 1 1 1 1 1 1 1 1 ...

# Now we can add the `group` variable to the dataset.
data_ex_1 <- cbind(data_ex_1, group = group)

# Inspect the head of the data.
head(data_ex_1)
```



```
##          inf          sim          voc          com          p_com          p_arr          b_des          o_ass group
## 1 10.611565 10.037764  9.130394 11.283947 12.688104 14.753140  7.705091  8.936030 manic
## 2  3.934381  7.689681  2.964765  7.333193 11.178762 10.065857  6.005804  7.658062 manic
## 3  8.234517 10.474990  7.648771 11.680683  9.500099  7.070482 11.253965 12.700114 manic
## 4 17.341116 18.674477 11.327905 12.748632 12.170414 12.589906 12.265402 14.160554 manic
## 5 10.254932 11.662189  7.893027  6.740234  8.954935  4.073957  5.995267 10.199569 manic
## 6 11.477607 14.029962 10.700104 11.057523  8.102913  6.095811 13.159574  6.806254 manic
```

```
# Inspect the tail of the data.
tail(data_ex_1)
```

```
##          inf          sim          voc          com          p_com          p_arr          b_des          o_ass group
## 395  9.645569 10.603463  8.045378  9.272881  9.107972 10.680583 10.918464  7.269432 norming
## 396  9.685306  9.747118  5.585436  8.525569  9.477196 17.859655 11.879686  9.528950 norming
## 397 15.868305 15.255730 15.334440 14.216598 13.083955 13.440027 12.467222 10.008781 norming
## 398  7.830651  5.914111  6.729447  4.816376  7.086313  2.400862  4.919694  1.856532 norming
## 399  5.884157  7.718305  3.908084  7.182868  5.671944 11.791069 10.473924  5.922491 norming
## 400 13.042982 13.321951  9.408113 14.672445 10.559942  8.451276  7.966637 12.787075 norming
```

With the data in hand, we can now perform the test for configural measurement invariance. This time we use the full data and make use of the `group` argument in `lavaan`. Note that the model syntax remains the same.

```
# Fit the model using the full data.
model_ex_1_configural_data_fit <- sem(model_ex_1, data_ex_1, group = "group")

# Model summary.
summary(model_ex_1_configural_data_fit, standardized = TRUE)
```

```
## lavaan 0.6-12 ended normally after 101 iterations
##
##      Estimator                      ML
##      Optimization method          NLMINB
##      Number of model parameters          50
##
##      Number of observations per group:
##      manic                          100
##      norming                        300
##
## Model Test User Model:
##
##      Test statistic                  97.221
##      Degrees of freedom              38
##      P-value (Chi-square)            0.000
##      Test statistic for each group:
##      manic                          43.234
##      norming                        53.987
##
## Parameter Estimates:
##
##      Standard errors                Standard
##      Information                    Expected
##      Information saturated (h1) model  Structured
##
##
## Group 1 [manic]:
##
```

```

## Latent Variables:
##           Estimate Std.Err z-value P(>|z|) Std.lv Std.all
## verbal =~
##   inf           1.000           2.280 0.725
##   sim           1.331 0.149 8.916 0.000 3.034 0.911
##   voc           1.222 0.146 8.386 0.000 2.786 0.853
##   com           1.125 0.137 8.207 0.000 2.564 0.835
## visual =~
##   p_com         1.000           1.905 0.786
##   p_arr         1.251 0.205 6.113 0.000 2.383 0.637
##   b_des         1.124 0.184 6.099 0.000 2.141 0.636
##   o_ass         1.247 0.187 6.682 0.000 2.376 0.694
##
## Covariances:
##           Estimate Std.Err z-value P(>|z|) Std.lv Std.all
## verbal ~~
##   visual        3.594 0.753 4.774 0.000 0.827 0.827
##
## Intercepts:
##           Estimate Std.Err z-value P(>|z|) Std.lv Std.all
##   .inf          10.240 0.314 32.574 0.000 10.240 3.257
##   .sim           11.875 0.333 35.643 0.000 11.875 3.564
##   .voc           10.082 0.327 30.852 0.000 10.082 3.085
##   .com           10.095 0.307 32.861 0.000 10.095 3.286
##   .p_com         10.773 0.242 44.457 0.000 10.773 4.446
##   .p_arr         10.677 0.374 28.562 0.000 10.677 2.856
##   .b_des         10.081 0.337 29.945 0.000 10.081 2.995
##   .o_ass         10.206 0.343 29.789 0.000 10.206 2.979
##   verbal         0.000           0.000 0.000
##   visual         0.000           0.000 0.000
##
## Variances:
##           Estimate Std.Err z-value P(>|z|) Std.lv Std.all
##   .inf           4.683 0.731 6.408 0.000 4.683 0.474
##   .sim           1.895 0.462 4.101 0.000 1.895 0.171
##   .voc           2.915 0.538 5.414 0.000 2.915 0.273
##   .com           2.863 0.506 5.657 0.000 2.863 0.303
##   .p_com         2.243 0.455 4.925 0.000 2.243 0.382
##   .p_arr         8.297 1.341 6.186 0.000 8.297 0.594
##   .b_des         6.748 1.090 6.192 0.000 6.748 0.595
##   .o_ass         6.090 1.039 5.861 0.000 6.090 0.519
##   verbal         5.199 1.269 4.097 0.000 1.000 1.000
##   visual         3.629 0.834 4.351 0.000 1.000 1.000
##
##
## Group 2 [norming]:
##
## Latent Variables:
##           Estimate Std.Err z-value P(>|z|) Std.lv Std.all
## verbal =~
##   inf           1.000           2.539 0.800

```

```
##      sim            1.011    0.061   16.611    0.000    2.568    0.865
##      voc            1.077    0.064   16.714    0.000    2.736    0.870
##      com            0.695    0.061   11.324    0.000    1.765    0.633
##  visual =~
##      p_com          1.000
##      p_arr          0.549    0.090    6.106    0.000    1.316    0.390
##      b_des          1.070    0.095   11.292    0.000    2.565    0.753
##      o_ass          1.021    0.092   11.095    0.000    2.447    0.736
##
## Covariances:
##      Estimate Std.Err z-value P(>|z|) Std.lv Std.all
##  verbal ~~
##      visual      4.701    0.595    7.905    0.000    0.772    0.772
##
## Intercepts:
##      Estimate Std.Err z-value P(>|z|) Std.lv Std.all
##      .inf      9.810    0.183   53.545    0.000    9.810    3.091
##      .sim      10.106    0.171   58.946    0.000   10.106    3.403
##      .voc      9.589    0.182   52.797    0.000    9.589    3.048
##      .com      10.047    0.161   62.393    0.000   10.047    3.602
##      .p_com     10.053    0.192   52.411    0.000   10.053    3.026
##      .p_arr      9.985    0.195   51.269    0.000    9.985    2.960
##      .b_des      9.683    0.197   49.241    0.000    9.683    2.843
##      .o_ass      10.052    0.192   52.353    0.000   10.052    3.023
##      verbal      0.000
##      visual      0.000
##
## Variances:
##      Estimate Std.Err z-value P(>|z|) Std.lv Std.all
##      .inf      3.621    0.368    9.828    0.000    3.621    0.360
##      .sim      2.222    0.273    8.130    0.000    2.222    0.252
##      .voc      2.409    0.303    7.946    0.000    2.409    0.243
##      .com      4.664    0.410   11.365    0.000    4.664    0.600
##      .p_com     5.288    0.561    9.426    0.000    5.288    0.479
##      .p_arr     9.647    0.818   11.796    0.000    9.647    0.848
##      .b_des     5.021    0.568    8.833    0.000    5.021    0.433
##      .o_ass     5.072    0.553    9.176    0.000    5.072    0.459
##      verbal     6.449    0.798    8.080    0.000    1.000    1.000
##      visual     5.750    0.868    6.621    0.000    1.000    1.000
```

```
# Fit measures.
fitMeasures(model_ex_1_configural_data_fit, fit.measures = fit_indices)
```

```
##      chisq      df      pvalue      cfi      tli      rmsea rmsea.pvalue
##      97.221    38.000      0.000      0.962      0.943      0.088      0.003
##      srmr
##      0.040
```

```
# Save the model plots for each groups as a list with two elements.
plots_configural_data <- semPaths(
  model_ex_1_configural_data_fit,
  what = "paths",
  whatLabels = "est",
```

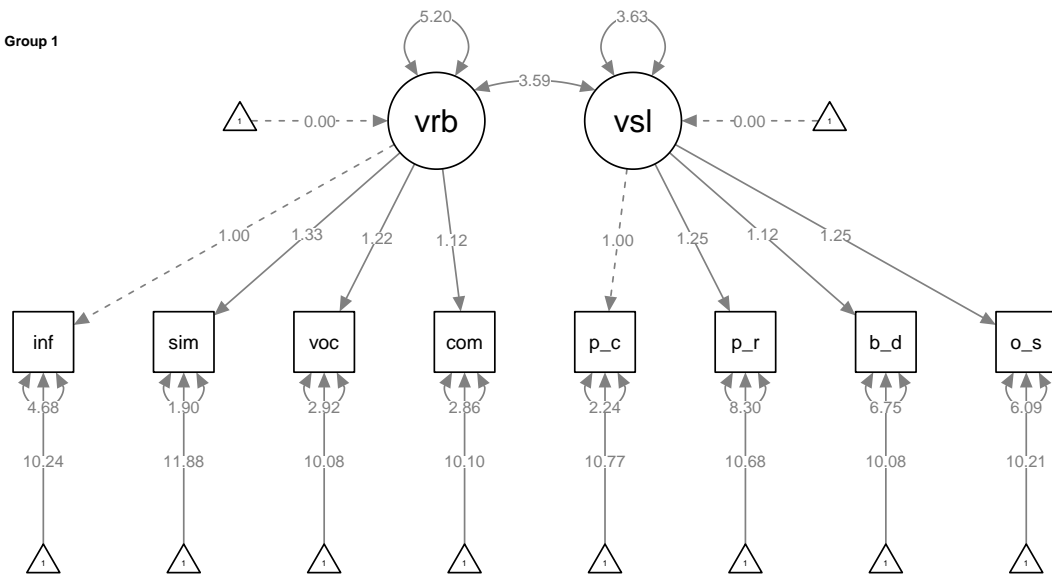
```

    DoNotPlot = TRUE,
    ask = FALSE,
    title = FALSE
  )

# Plot the model for the first group.
plot(plots_configural_data[[1]])
title("Configural MI (data) | Group 1", adj = 0)

```

Configural MI (data) | Group 1

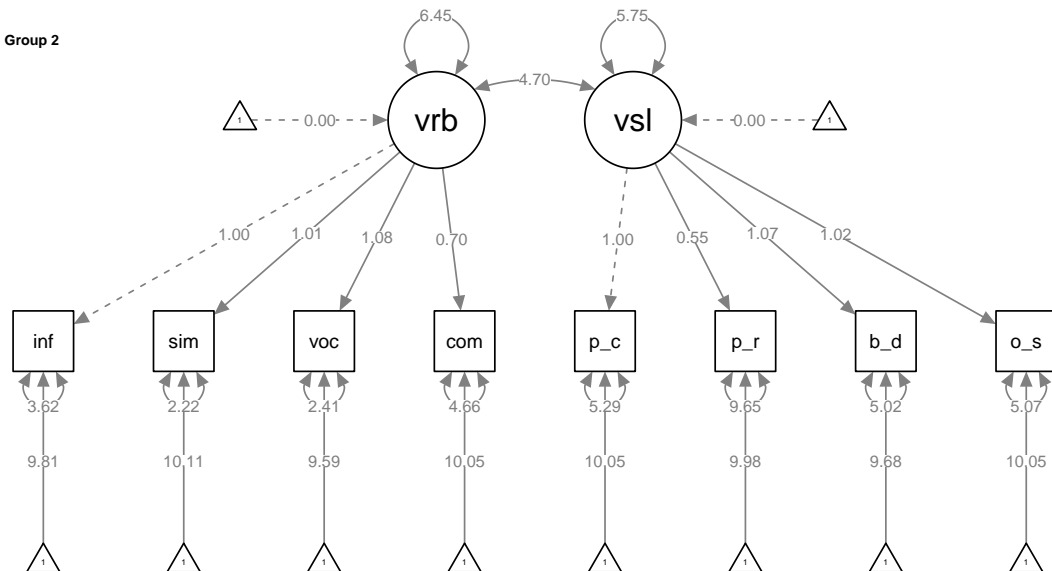


```

# Plot the model for the second group.
plot(plots_configural_data[[2]])
title("Configural MI (data) | Group 2", adj = 0)

```

Configural MI (data) | Group 2



Exercise 2

Part 1. Investigating longitudinal measurement invariance.

In this part of *Exercise 2*, you are going to investigate measurement invariance using longitudinal data. This exercise differs a bit from the lecture, in the sense that we will not deploy measurement invariance tests to check whether we can validly compare models parameters across groups. *Instead, we test for measurement invariance to understand whether we can validly compare model parameters across time.* In other words, we are interested to understand whether our construct is measurement invariant from one measurement occasion to the other.

Since testing for measurement invariance for longitudinal data is slightly more involved than what we did during *Exercise 1*, I recommend you to take a look at *Chapter 2* from Newsom (2015) (i.e., attached on Canvas under the *References* heading for the current practical). For a quick overview of the parametrization required, you can take a look at *Figure 3*. The relationships depicted in *Figure 3* are also described below:

- **Weak measurement invariance:** when loadings are equal over time but intercepts, unique variances, latent means, and latent variances vary over time.
- **Strong measurement invariance:** when loadings and intercepts do not vary but unique variances, latent means, and latent variances vary over time.
- **Strict measurement invariance:** when loadings, intercepts, and measurement residuals are equal over time.
- **Structural invariance:** when factor means, factor variances, loadings, intercepts, and measurement residuals are equal over time.

Key points to keep in mind for this exercise:

- During *Exercise 1* we applied the constraints across groups (e.g., a loading in group one was constrained to be equal with the corresponding loading in group two).
- However, when we check for measurement invariance for longitudinal data, we think of the measurement occasions as our “groups”. In this case, we
 1. fit the latent construct at both measurement occasions in the same model,
 2. and constrain the parameters to be equal across time points (e.g., a loading for the latent construct at time point one constrained to be equal to the corresponding loading for the latent construct at time point two).
- So, the same ideas that you discussed during the lecture apply, but instead of fitting one model per group, you now fit a single model (i.e., with a latent construct per time point), and you constrain the parameters to be equal across time points.

The data you will use consists of a set of three items (i.e., `w1vst1`, `w1vst2`, `w1vst3`, `w2vst1`, `w2vst2`, and `w2vst3`) measured at two time points (i.e., `w1` for wave one and `w2` for wave two), with a total sample size of $N = 574$. You can find the data in the folder for this practical on Canvas, and the baseline model that you will investigate is graphically represented in *Figure 4*.

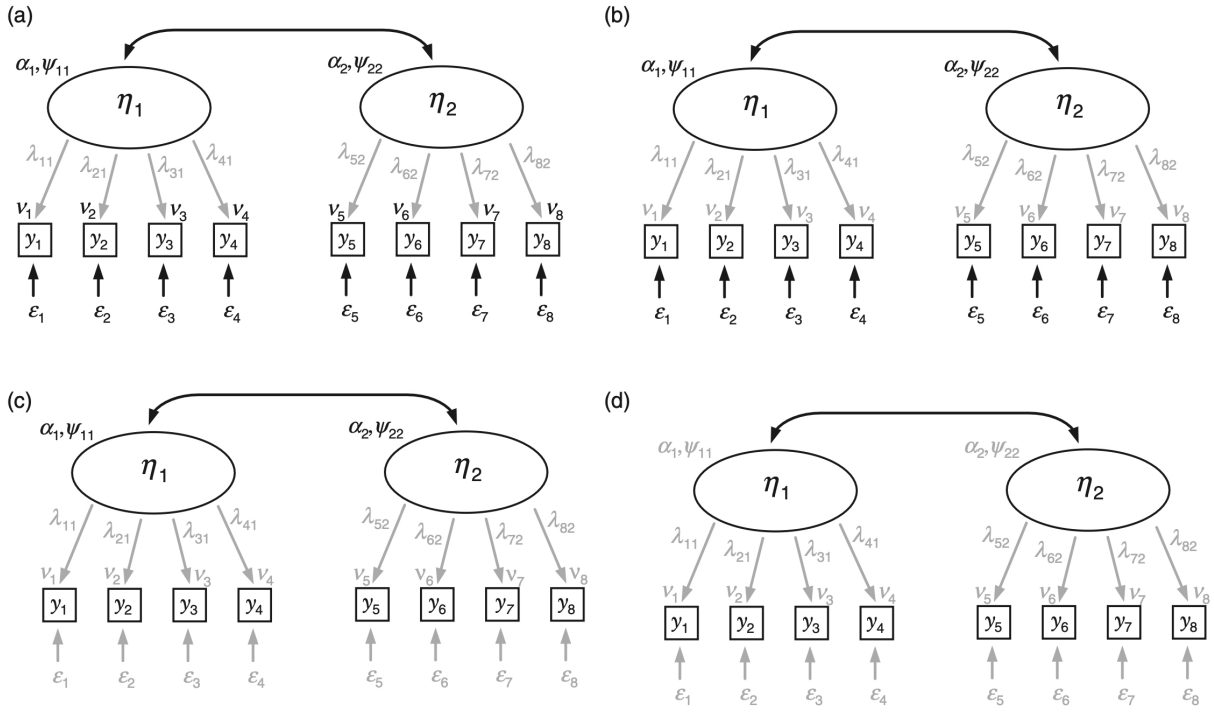


Figure 2.1 Graphic Depiction of Meredith's Factorial Invariance Definitions: (a) weak invariance; (b) strong invariance; (c) strict invariance; (d) structural invariance. Note: grayed lines and symbols represent parameters that are equal over time, where it is assumed that the equality is only between the longitudinal counterparts of each (e.g., $\lambda_{11} = \lambda_{52}$ or $v_1 = v_5$).

Figure 3: Reproduction of *Figure 2.1* from Newsom (2015).

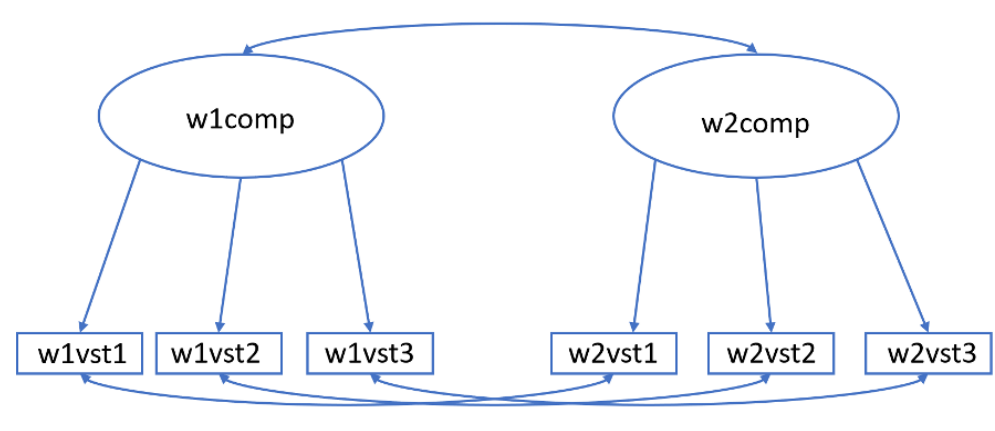


Figure 4: Model example for investigating longitudinal measurement invariance.

Start this exercise by loading the dataset `socex1.dat` in R and adding the following variable names to the data.

Set the working directory to the location where your data file has been downloaded and load the data.

```
# For example.
setwd("/Users/mihai/Downloads")

# Load data.
data_ex_2 <- read.csv("socex1.dat")

# Inspect the data.
View(data_ex_2)
```

Set the variable names.

```
# Variable names.
variable_ex_2_names <- c(
  "w1vst1", "w1vst2", "w1vst3", "w2vst1", "w2vst2",
  "w2vst3", "w3vst1", "w3vst2", "w3vst3", "w1unw1", "w1unw2", "w1unw3",
  "w2unw1", "w2unw2", "w2unw3", "w3unw1", "w3unw2", "w3unw3", "w1dboth",
  "w1dsad", "w1dblues", "w1ddep", "w2dboth", "w2dsad", "w2dblues", "w2ddep",
  "w3dboth", "w3dsad", "w3dblues", "w3ddep", "w1marr2", "w1happy", "w1enjoy",
  "w1satis", "w1joyful", "w1please", "w2happy", "w2enjoy", "w2satis", "w2joyful",
  "w2please", "w3happy", "w3enjoy", "w3satis", "w3joyful", "w3please", "w1lea",
  "w2lea", "w3lea"
)

# Set the names.
names(data_ex_2) <- variable_ex_2_names

# List variables.
str(data_ex_2)
```

```
## 'data.frame': 574 obs. of 49 variables:
## $ w1vst1 : int 0 3 2 2 3 4 1 3 4 1 ...
## $ w1vst2 : int 1 3 2 2 2 4 1 2 4 0 ...
## $ w1vst3 : int 0 4 3 3 2 4 0 2 4 0 ...
## $ w2vst1 : int 3 3 1 1 2 4 3 3 3 2 ...
## $ w2vst2 : int 3 3 0 2 2 4 1 2 2 3 ...
## $ w2vst3 : int 2 3 2 2 2 4 2 2 2 3 ...
## $ w3vst1 : int 3 2 2 2 3 4 2 2 2 3 ...
## $ w3vst2 : int 2 3 2 2 3 4 3 2 3 3 ...
## $ w3vst3 : int 2 3 1 2 2 4 2 2 3 2 ...
## $ w1unw1 : int 2 1 2 4 2 0 2 4 3 4 ...
## $ w1unw2 : int 2 3 1 3 4 2 2 3 1 2 ...
## $ w1unw3 : int 3 2 1 3 3 1 2 3 3 3 ...
## $ w2unw1 : int 4 3 1 3 2 2 3 3 2 2 ...
## $ w2unw2 : int 4 2 3 2 3 1 2 3 3 3 ...
## $ w2unw3 : int 4 3 2 1 2 1 3 4 3 2 ...
## $ w3unw1 : int 2 2 2 3 2 2 2 4 2 2 ...
## $ w3unw2 : int 3 3 2 4 3 1 2 4 3 3 ...
## $ w3unw3 : int 3 2 2 2 2 1 1 3 2 2 ...
```

```
## $ w1dboth : int 0 0 0 2 1 0 0 0 0 0 ...
## $ w1dsad : int 0 1 0 0 2 0 0 2 0 0 ...
## $ w1dblues: int 1 1 0 0 1 0 1 0 0 0 ...
## $ w1ddep : int 0 1 0 0 1 0 2 1 0 0 ...
## $ w2dboth : int 0 0 1 1 0 0 0 2 0 0 ...
## $ w2dsad : int 1 1 1 0 1 0 0 0 1 1 ...
## $ w2dblues: int 0 2 1 0 0 0 0 2 0 0 ...
## $ w2ddep : int 1 2 0 3 0 0 0 3 1 0 ...
## $ w3dboth : int 0 1 0 1 0 1 0 2 0 1 ...
## $ w3dsad : int 0 0 0 1 0 0 0 2 0 1 ...
## $ w3dblues: int 1 2 0 2 0 1 0 1 0 1 ...
## $ w3ddep : int 0 2 0 0 2 0 0 1 0 0 ...
## $ w1marr2 : int 1 1 1 1 0 1 1 1 1 0 ...
## $ w1happy : int 3 3 3 2 2 5 2 2 2 4 ...
## $ w1enjoy : int 3 3 2 3 3 5 3 2 4 3 ...
## $ w1satis : int 3 3 3 3 3 4 2 2 4 3 ...
## $ w1joyful: int 3 3 2 3 2 4 3 2 3 3 ...
## $ w1please: int 3 2 3 4 2 4 2 1 3 3 ...
## $ w2happy : num 2.9 3.9 1.9 2.9 1.9 4.9 2.9 1.9 3.9 2.9 ...
## $ w2enjoy : num 2.9 2.9 2.9 1.9 1.9 3.9 2.9 2.9 2.9 3.9 ...
## $ w2satis : num 3.9 3.9 2.9 2.9 1.9 3.9 2.9 2.9 1.9 3.9 ...
## $ w2joyful: num 2.9 2.9 2.9 3.9 2.9 3.9 2.9 1.9 3.9 2.9 ...
## $ w2please: num 2.9 2.9 2.9 2.9 1.9 3.9 1.9 1.9 3.9 2.9 ...
## $ w3happy : num 2.8 2.8 2.8 2.8 2.8 3.8 2.8 1.8 2.8 3.8 ...
## $ w3enjoy : num 1.8 3.8 2.8 3.8 2.8 3.8 2.8 2.8 3.8 3.8 ...
## $ w3satis : num 1.8 2.8 1.8 2.8 1.8 4.8 2.8 1.8 2.8 2.8 ...
## $ w3joyful: num 2.8 2.8 1.8 2.8 2.8 4.8 1.8 1.8 2.8 2.8 ...
## $ w3please: num 2.8 2.8 2.8 2.8 2.8 3.8 1.8 1.8 3.8 2.8 ...
## $ w1lea : int 0 0 0 0 0 0 0 0 0 0 ...
## $ w2lea : int 0 0 0 0 0 0 0 0 0 0 ...
## $ w3lea : int 0 0 0 0 1 0 0 1 0 0 ...
```

Specify which fit measures we are interested in:

```
# Fit indices to print.
fit_indices <- c("chisq", "df", "pvalue", "cfi", "tli", "rmsea", "rmsea.pvalue", "srmr")
```

- a. Investigate *configural* measurement invariance and also estimate the means of the latent variables. Report and interpret the model fit.

Contrary to *Exercise 1*, we now have to set the constraints manually. Few things to note:

- we use the default marker variable approach for setting the scales of latent variables `w1comp` and `w2comp`
- since we are going to estimate the means of the latent variables we must fix the first intercept at each time point to 0 for model identification

```
# Model syntax.
model_ex_2_configural <- "
  # Measurement part.
  w1comp =~ w1vst1 + w1vst2 + w1vst3
  w2comp =~ w2vst1 + w2vst2 + w2vst3

  # Covariance between latent variables.
```



```

w2comp ~~ w1comp

# Covariances between residuals across time.
w1vst1 ~~ w2vst1
w1vst2 ~~ w2vst2
w1vst3 ~~ w2vst3

# Fix first intercept at each time point to 0 for identification.
w1vst1 ~ 0
w2vst1 ~ 0

# Freely estimate means of latent variables.
w1comp ~ 1
w2comp ~ 1
"

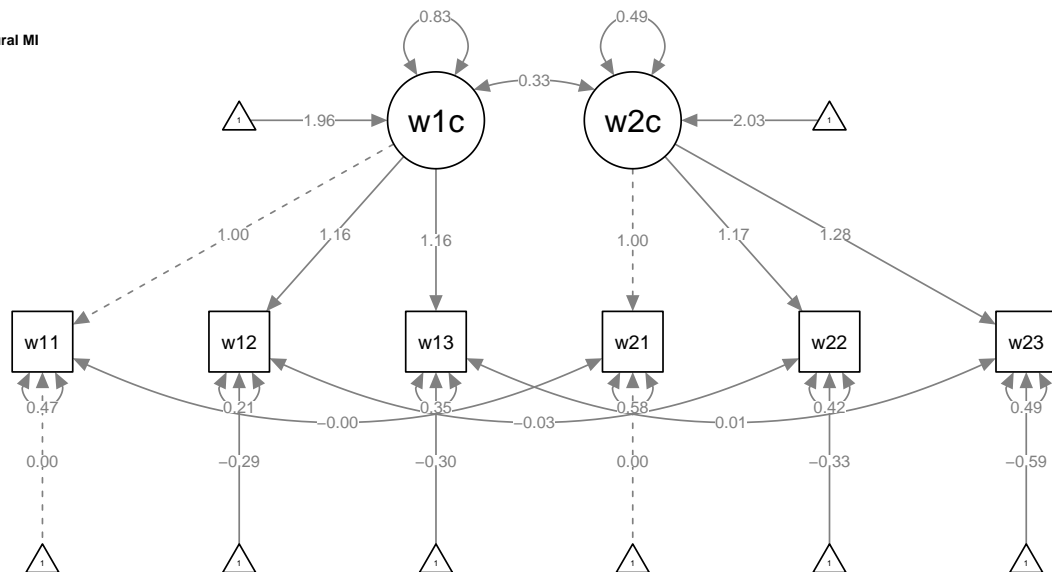
# Fit the model.
model_ex_2_configural_fit <- sem(model_ex_2_configural, data_ex_2)

# Visualize the model.
semPaths(model_ex_2_configural_fit, what = "paths", whatLabels = "est")

# Add a title to the plot.
title("Longitudinal Configural MI", adj = 0)

```

Longitudinal Configural MI



```

# Model summary.
summary(model_ex_2_configural_fit, standardized = TRUE)

```

```

## lavaan 0.6-12 ended normally after 47 iterations
##
##      Estimator              ML
##      Optimization method    NLMINB
##      Number of model parameters    22
##

```

```

## Number of observations                    574
##
## Model Test User Model:
##
## Test statistic                          9.911
## Degrees of freedom                      5
## P-value (Chi-square)                   0.078
##
## Parameter Estimates:
##
## Standard errors                        Standard
## Information                          Expected
## Information saturated (h1) model      Structured
##
## Latent Variables:
##      Estimate Std.Err z-value P(>|z|) Std.lv Std.all
## w1comp =~
## w1vst1      1.000
## w1vst2      1.159    0.048   24.360   0.000    1.059    0.917
## w1vst3      1.156    0.049   23.659   0.000    1.056    0.874
## w2comp =~
## w2vst1      1.000
## w2vst2      1.173    0.081   14.419   0.000    0.819    0.784
## w2vst3      1.284    0.088   14.556   0.000    0.897    0.788
##
## Covariances:
##      Estimate Std.Err z-value P(>|z|) Std.lv Std.all
## w1comp ~~
## w2comp      0.331    0.039    8.442   0.000    0.519    0.519
## .w1vst1 ~~
## .w2vst1     -0.001    0.026   -0.045   0.965   -0.001   -0.002
## .w1vst2 ~~
## .w2vst2     -0.030    0.021   -1.417   0.156   -0.030   -0.102
## .w1vst3 ~~
## .w2vst3      0.013    0.025    0.509   0.610    0.013    0.032
##
## Intercepts:
##      Estimate Std.Err z-value P(>|z|) Std.lv Std.all
## .w1vst1      0.000
## .w2vst1      0.000
## w1comp      1.960    0.048   41.134   0.000    2.146    2.146
## w2comp      2.031    0.043   47.003   0.000    2.909    2.909
## .w1vst2     -0.293    0.101   -2.904   0.004   -0.293   -0.254
## .w1vst3     -0.297    0.104   -2.851   0.004   -0.297   -0.246
## .w2vst2     -0.333    0.172   -1.944   0.052   -0.333   -0.319
## .w2vst3     -0.589    0.186   -3.165   0.002   -0.589   -0.518
##
## Variances:
##      Estimate Std.Err z-value P(>|z|) Std.lv Std.all
## .w1vst1      0.469    0.034   13.798   0.000    0.469    0.360
## .w1vst2      0.212    0.029    7.337   0.000    0.212    0.159

```

##	.w1vst3	0.346	0.033	10.511	0.000	0.346	0.237
##	.w2vst1	0.585	0.043	13.557	0.000	0.585	0.545
##	.w2vst2	0.419	0.042	9.946	0.000	0.419	0.385
##	.w2vst3	0.490	0.049	9.900	0.000	0.490	0.379
##	w1comp	0.834	0.074	11.206	0.000	1.000	1.000
##	w2comp	0.488	0.059	8.261	0.000	1.000	1.000

```
# Fit measures.
fitMeasures(model_ex_2_configural_fit, fit.measures = fit_indices)
```

##	chisq	df	pvalue	cfi	tli	rmsea	rmsea.pvalue
##	9.911	5.000	0.078	0.997	0.991	0.041	0.591
##	srmr						
##	0.018						

The configural measurement invariance model fits the data well and all loadings are significant. The other fit indices (e.g., *RMSEA* and *CFI*) also show indicate good fit. The estimated means of the latent variables are 1.960 for *w1comp* and 2.031 for *w2comp*. Note that we cannot yet compare these means.

- b. Investigate *weak* or *metric* measurement invariance. Test if the additional constraints of the weak invariance model do not significantly worsen model fit in comparison to the configural model. Report and interpret the results.

For the metric measurement invariance, we need to constrain the loadings to be equal across time points. To do this, we can use labels. Note that the marker items do not need equality constraint, as those loadings are already set to 1 for identification purposes. Just as for point (a), we leave the mean structure unconstrained.

```
# Model syntax.
model_ex_2_weak <- "
  # Measurement part.
  w1comp =~ w1vst1 + a * w1vst2 + b * w1vst3
  w2comp =~ w2vst1 + a * w2vst2 + b * w2vst3

  # Covariance between latent variables.
  w2comp ~~ w1comp

  # Covariances between residuals across time.
  w1vst1 ~~ w2vst1
  w1vst2 ~~ w2vst2
  w1vst3 ~~ w2vst3

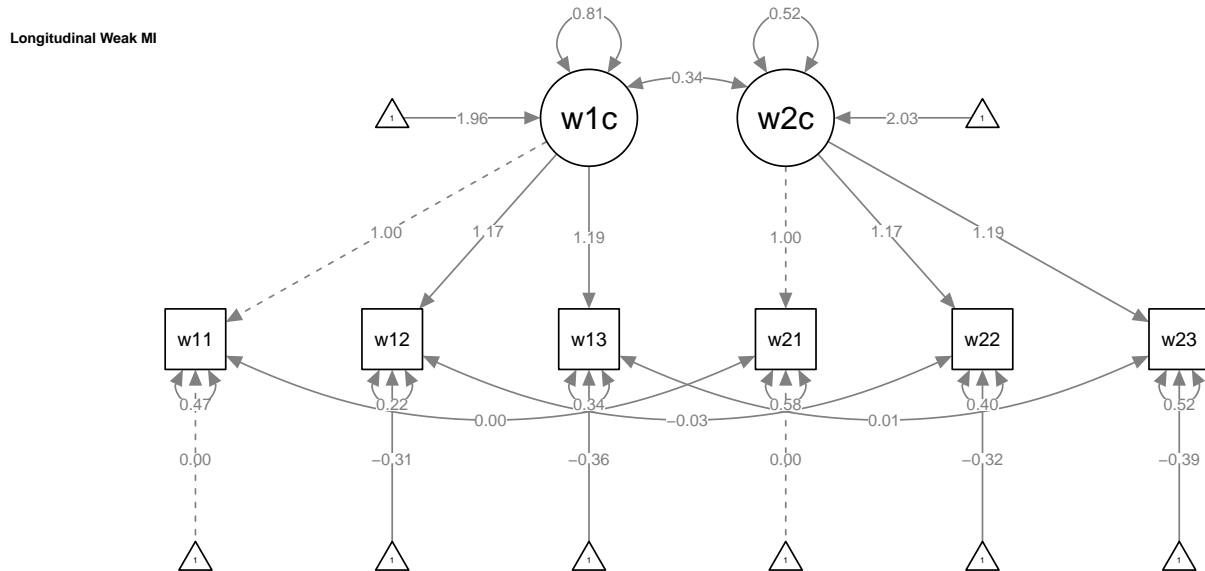
  # Fix first intercept at each time point to 0 for identification.
  w1vst1 ~ 0
  w2vst1 ~ 0

  # Freely estimate means of latent variables.
  w1comp ~ 1
  w2comp ~ 1
"

# Fit the model.
model_ex_2_weak_fit <- sem(model_ex_2_weak, data_ex_2)
```

```
# Visualize the model.
semPaths(model_ex_2_weak_fit, what = "paths", whatLabels = "est")

# Add a title to the plot.
title("Longitudinal Weak MI", adj = 0)
```



```
# Model summary.
summary(model_ex_2_weak_fit, standardized = TRUE)
```

```
## lavaan 0.6-12 ended normally after 38 iterations
##
##   Estimator           ML
##   Optimization method  NLMINB
##   Number of model parameters    22
##   Number of equality constraints    2
##
##   Number of observations    574
##
## Model Test User Model:
##
##   Test statistic    12.077
##   Degrees of freedom    7
##   P-value (Chi-square)    0.098
##
## Parameter Estimates:
##
##   Standard errors    Standard
##   Information    Expected
##   Information saturated (h1) model    Structured
##
## Latent Variables:
##
##           Estimate Std.Err z-value P(>|z|) Std.lv Std.all
## w1comp =~
##   w1vst1    1.000    0.902    0.796
```

```
##      w1vst2      (a)      1.166      0.041      28.422      0.000      1.051      0.914
##      w1vst3      (b)      1.188      0.043      27.672      0.000      1.071      0.879
##      w2comp =~
##      w2vst1              1.000              0.721      0.689
##      w2vst2      (a)      1.166      0.041      28.422      0.000      0.840      0.799
##      w2vst3      (b)      1.188      0.043      27.672      0.000      0.856      0.764
##
## Covariances:
##              Estimate Std.Err z-value P(>|z|) Std.lv Std.all
##      w1comp ~~
##      w2comp      0.335      0.039      8.613      0.000      0.516      0.516
##      .w1vst1 ~~
##      .w2vst1      0.001      0.026      0.025      0.980      0.001      0.001
##      .w1vst2 ~~
##      .w2vst2     -0.032      0.021     -1.513      0.130     -0.032     -0.110
##      .w1vst3 ~~
##      .w2vst3      0.014      0.025      0.533      0.594      0.014      0.032
##
## Intercepts:
##              Estimate Std.Err z-value P(>|z|) Std.lv Std.all
##      .w1vst1      0.000              0.000      0.000
##      .w2vst1      0.000              0.000      0.000
##      w1comp      1.960      0.047     41.425      0.000      2.173      2.173
##      w2comp      2.031      0.044     46.513      0.000      2.819      2.819
##      .w1vst2     -0.305      0.089     -3.424      0.001     -0.305     -0.266
##      .w1vst3     -0.359      0.094     -3.823      0.000     -0.359     -0.295
##      .w2vst2     -0.319      0.095     -3.363      0.001     -0.319     -0.303
##      .w2vst3     -0.393      0.100     -3.948      0.000     -0.393     -0.351
##
## Variances:
##              Estimate Std.Err z-value P(>|z|) Std.lv Std.all
##      .w1vst1      0.471      0.034     13.997      0.000      0.471      0.367
##      .w1vst2      0.218      0.028      7.775      0.000      0.218      0.165
##      .w1vst3      0.338      0.033     10.343      0.000      0.338      0.227
##      .w2vst1      0.576      0.042     13.777      0.000      0.576      0.526
##      .w2vst2      0.400      0.038     10.505      0.000      0.400      0.362
##      .w2vst3      0.522      0.044     11.869      0.000      0.522      0.416
##      w1comp      0.814      0.068     11.951      0.000      1.000      1.000
##      w2comp      0.519      0.046     11.219      0.000      1.000      1.000
```

```
# Fit measures.
fitMeasures(model_ex_2_weak_fit, fit.measures = fit_indices)
```

```
##      chisq      df      pvalue      cfi      tli      rmsea rmsea.pvalue
##      12.077      7.000      0.098      0.997      0.994      0.036      0.728
##      srmr
##      0.025
```

Perform a LRT between the week model and the configural model.

```
# Perform LRT.
anova(model_ex_2_weak_fit, model_ex_2_configural_fit)
```

```
## Chi-Squared Difference Test
##
##           Df      AIC      BIC   Chisq Chisq diff Df diff Pr(>Chisq)
## model_ex_2_configural_fit  5 8864.6 8960.3  9.9108
## model_ex_2_weak_fit       7 8862.7 8949.8 12.0767      2.166      2      0.3386
```

```
# Combine the fit measures for the configural and weak model.
fit_measures_all_ex_2 <- rbind(
  configural = fitMeasures(model_ex_2_configural_fit, fit_indices),
  weak = fitMeasures(model_ex_2_weak_fit, fit_indices)
)
```

```
# Print fit measures with four decimals.
print(round(fit_measures_all_ex_2, 4))
```

```
##           chisq df pvalue      cfi      tli  rmsea rmsea.pvalue      srmr
## configural  9.9108  5 0.0778 0.9971 0.9914 0.0414      0.5909 0.0181
## weak       12.0767  7 0.0981 0.9970 0.9937 0.0355      0.7284 0.0251
```

The results indicate good model fit with a $\chi^2 = 12.076$ with 7 degrees of freedom and a p -value = 0.098. The change in CFI between the weak model and the configural model is 0, and $RMSEA$ has dropped from 0.0414 to .0355, in accordance with rules of thumb. The χ^2 difference test between the two models (i.e., weak and configural) is also non-significant with p -value = 0.338, indicating that the constrained model does not significantly worsen the fit. Based on these results, we can conclude that we have support for weak longitudinal measurement invariance and continue testing for the next level of measurement invariance.

- c. Investigate *strong* or *scalar* measurement invariance, now with the necessary equality constraints on the intercepts. Test if the additional constraints of the strong invariance model do not significantly worsen model fit in comparison to the weak invariance model. Report and interpret the results.

For the scalar measurement invariance we need to constrain the intercepts to be equal across time. Since we are interested in the means of the latent variables, we must fix the intercept for the first indicator at each time point to 0 for identification purposes. Therefore, we can only constrain the remaining intercepts to be equal across time.

```
# Model syntax.
model_ex_2_strong <- "
  # Measurement part.
  w1comp =~ w1vst1 + a * w1vst2 + b * w1vst3
  w2comp =~ w2vst1 + a * w2vst2 + b * w2vst3

  # Covariance between latent variables.
  w2comp ~~ w1comp

  # Covariances between residuals across time.
  w1vst1 ~~ w2vst1
  w1vst2 ~~ w2vst2
  w1vst3 ~~ w2vst3

  # Fix first intercept at each time point to 0 for identification.
  w1vst1 ~ 0
```

```

w2vst1 ~ 0

# Constrain the remaining intercepts to be equal across time.
w1vst2 ~ c * 1
w1vst3 ~ d * 1
w2vst2 ~ c * 1
w2vst3 ~ d * 1

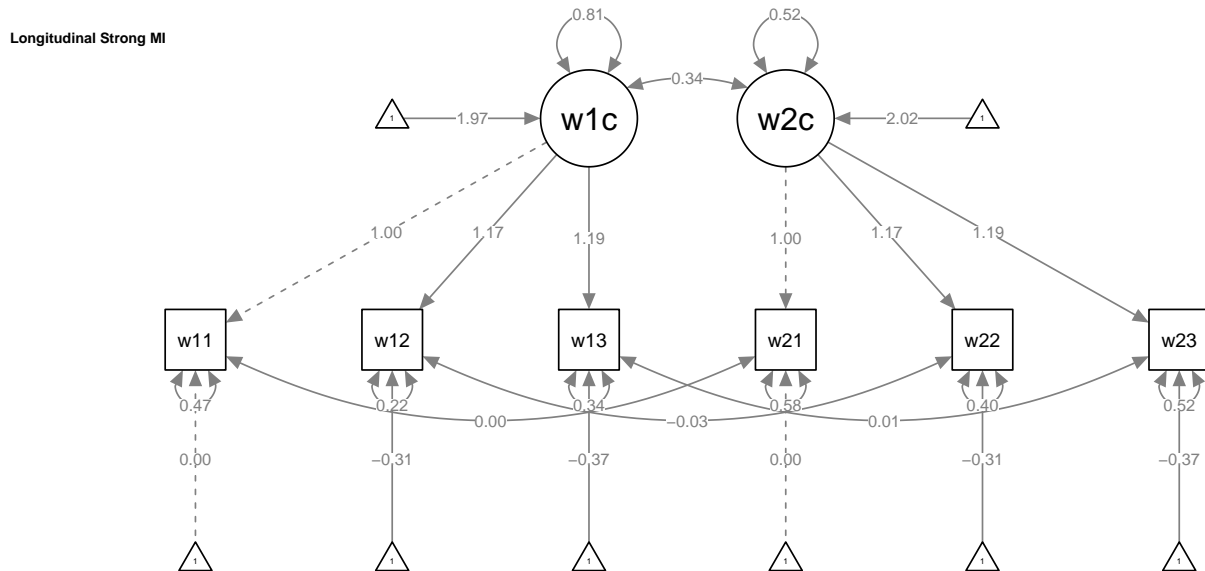
# Freely estimate means of latent variables.
w1comp ~ 1
w2comp ~ 1
"

# Fit the model.
model_ex_2_strong_fit <- sem(model_ex_2_strong, data_ex_2)

# Visualize the model.
semPaths(model_ex_2_strong_fit, what = "paths", whatLabels = "est")

# Add a title to the plot.
title("Longitudinal Strong MI", adj = 0)

```



```

# Model summary.
summary(model_ex_2_strong_fit, standardized = TRUE)

```

```

## lavaan 0.6-12 ended normally after 42 iterations
##
## Estimator ML
## Optimization method NLMINB
## Number of model parameters 22
## Number of equality constraints 4
##
## Number of observations 574
##

```

```

## Model Test User Model:
##
##   Test statistic                12.399
##   Degrees of freedom              9
##   P-value (Chi-square)           0.192
##
## Parameter Estimates:
##
##   Standard errors                Standard
##   Information                    Expected
##   Information saturated (h1) model Structured
##
## Latent Variables:
##
##           Estimate Std.Err z-value P(>|z|) Std.lv Std.all
##   w1comp =~
##     w1vst1      1.000
##     w1vst2      (a) 1.165    0.041  28.448    0.000    1.052    0.914
##     w1vst3      (b) 1.187    0.043  27.693    0.000    1.071    0.879
##   w2comp =~
##     w2vst1      1.000
##     w2vst2      (a) 1.165    0.041  28.448    0.000    0.840    0.799
##     w2vst3      (b) 1.187    0.043  27.693    0.000    0.855    0.764
##
## Covariances:
##
##           Estimate Std.Err z-value P(>|z|) Std.lv Std.all
##   w1comp ~~
##     w2comp      0.336    0.039    8.615    0.000    0.516    0.516
##   .w1vst1 ~~
##     .w2vst1      0.001    0.026    0.023    0.982    0.001    0.001
##   .w1vst2 ~~
##     .w2vst2     -0.032    0.021   -1.513    0.130   -0.032   -0.110
##   .w1vst3 ~~
##     .w2vst3      0.014    0.025    0.531    0.595    0.014    0.032
##
## Intercepts:
##
##           Estimate Std.Err z-value P(>|z|) Std.lv Std.all
##   .w1vst1      0.000
##   .w2vst1      0.000
##   .w1vst2      (c) -0.311    0.087   -3.600    0.000   -0.311   -0.271
##   .w1vst3      (d) -0.372    0.091   -4.083    0.000   -0.372   -0.305
##   .w2vst2      (c) -0.311    0.087   -3.600    0.000   -0.311   -0.296
##   .w2vst3      (d) -0.372    0.091   -4.083    0.000   -0.372   -0.332
##   w1comp      1.967    0.044  44.577    0.000    2.180    2.180
##   w2comp      2.023    0.038  52.736    0.000    2.806    2.806
##
## Variances:
##
##           Estimate Std.Err z-value P(>|z|) Std.lv Std.all
##   .w1vst1      0.471    0.034  13.994    0.000    0.471    0.367
##   .w1vst2      0.218    0.028    7.770    0.000    0.218    0.165
##   .w1vst3      0.338    0.033  10.356    0.000    0.338    0.228
##   .w2vst1      0.576    0.042  13.775    0.000    0.576    0.526

```



```
##      .w2vst2          0.400    0.038   10.501    0.000    0.400    0.362
##      .w2vst3          0.523    0.044   11.878    0.000    0.523    0.417
##      w1comp           0.814    0.068   11.956    0.000    1.000    1.000
##      w2comp           0.520    0.046   11.222    0.000    1.000    1.000
```

```
# Fit measures.
fitMeasures(model_ex_2_strong_fit, fit.measures = fit_indices)
```

```
##      chisq      df      pvalue      cfi      tli      rmsea rmsea.pvalue
##      12.399      9.000      0.192      0.998      0.997      0.026      0.886
##      srmr
##      0.025
```

Perform a LRT between the strong model and the weak model.

```
# Perform LRT.
anova(model_ex_2_strong_fit, model_ex_2_weak_fit)
```

```
## Chi-Squared Difference Test
##
##              Df      AIC      BIC  Chisq Chisq diff Df diff Pr(>Chisq)
## model_ex_2_weak_fit      7 8862.7 8949.8 12.077
## model_ex_2_strong_fit    9 8859.0 8937.4 12.399      0.32254      2      0.8511
```

```
# Combine the fit measures for the configural and weak model.
fit_measures_all_ex_2 <- rbind(
  fit_measures_all_ex_2,
  strong = fitMeasures(model_ex_2_strong_fit, fit_indices)
)

# Print fit measures with four decimals.
print(round(fit_measures_all_ex_2, 4))
```

```
##      chisq df pvalue      cfi      tli rmsea rmsea.pvalue      srmr
## configural  9.9108  5 0.0778 0.9971 0.9914 0.0414      0.5909 0.0181
## weak       12.0767  7 0.0981 0.9970 0.9937 0.0355      0.7284 0.0251
## strong     12.3993  9 0.1917 0.9980 0.9967 0.0257      0.8864 0.0253
```

The results indicate good model fit with a $\chi^2 = 12.399$ with 9 degrees of freedom and a p -value = 0.192. The change in CFI between the strong model and the weak model is 0.001, and $RMSEA$ has dropped from 0.0355 to 0.0257, in accordance with rules of thumb. The χ^2 difference test between the two models (i.e., strong and weak) is also non-significant with p -value = 0.851, indicating that the constrained model does not significantly worsen the fit. Based on these results, we can conclude that we have support for strong longitudinal measurement invariance and continue testing for the next level of measurement invariance.

Since we now have support for strong measurement invariance, we can validly compare the means of the latent variables. We can, for example, test the hypothesis that the latent variable means are equal across time. To do this, we re-estimate the strong (i.e., scalar) measurement invariance model, with the additional constrain that the latent variable means should be equal over time.

```
# Model syntax.
model_ex_2_strong_equal_latent_means <- "
  # Measurement part.
```

```

w1comp =~ w1vst1 + a * w1vst2 + b * w1vst3
w2comp =~ w2vst1 + a * w2vst2 + b * w2vst3

# Covariance between latent variables.
w2comp ~~ w1comp

# Covariances between residuals across time.
w1vst1 ~~ w2vst1
w1vst2 ~~ w2vst2
w1vst3 ~~ w2vst3

# Fix first intercept at each time point to 0 for identification.
w1vst1 ~ 0
w2vst1 ~ 0

# Constrain the remaining intercepts to be equal across time.
w1vst2 ~ c * 1
w1vst3 ~ d * 1
w2vst2 ~ c * 1
w2vst3 ~ d * 1

# Freely estimate means of latent variables.
w1comp ~ h * 1
w2comp ~ h * 1
"

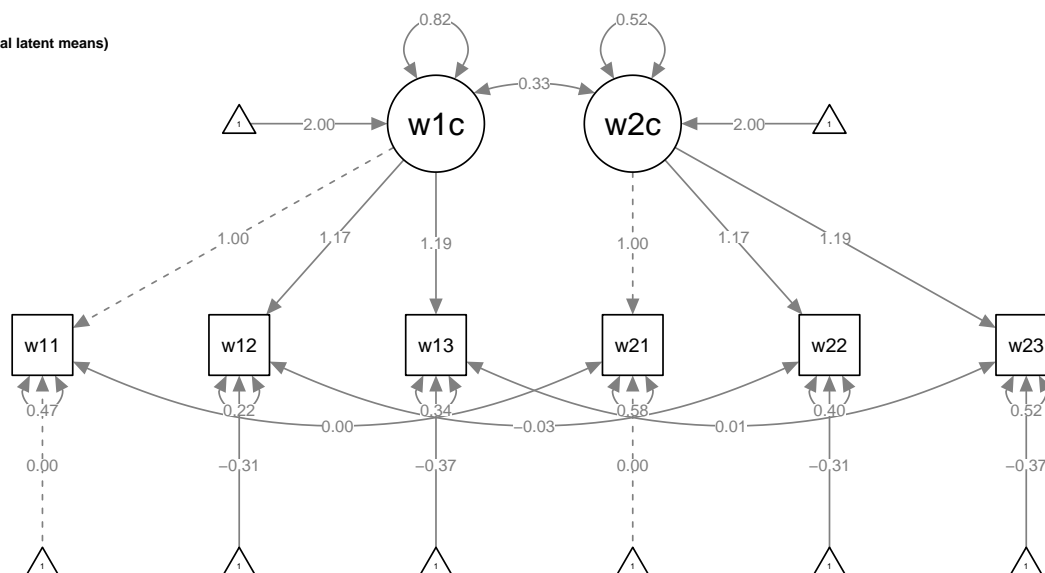
# Fit the model.
model_ex_2_strong_equal_latent_means_fit <- sem(model_ex_2_strong_equal_latent_means, data_ex_2)

# Visualize the model.
semPaths(model_ex_2_strong_equal_latent_means_fit, what = "paths", whatLabels = "est")

# Add a title to the plot.
title("Long. Strong MI (equal latent means)", adj = 0)

```

Long. Strong MI (equal latent means)



```
# Model summary.
summary(model_ex_2_strong_equal_latent_means_fit, standardized = TRUE)
```

lavaan 0.6-12 ended normally after 42 iterations

##

Estimator ML

Optimization method NLMINB

Number of model parameters 22

Number of equality constraints 5

##

Number of observations 574

##

Model Test User Model:

##

Test statistic 14.439

Degrees of freedom 10

P-value (Chi-square) 0.154

##

Parameter Estimates:

##

Standard errors Standard

Information Expected

Information saturated (h1) model Structured

##

Latent Variables:

	Estimate	Std.Err	z-value	P(> z)	Std.lv	Std.all
--	----------	---------	---------	---------	--------	---------

w1comp =~

w1vst1	1.000				0.903	0.796
--------	-------	--	--	--	-------	-------

w1vst2	(a)	1.165	0.041	28.446	0.000	1.052	0.914
--------	-----	-------	-------	--------	-------	-------	-------

w1vst3	(b)	1.187	0.043	27.693	0.000	1.072	0.879
--------	-----	-------	-------	--------	-------	-------	-------

w2comp =~

w2vst1		1.000				0.721	0.689
--------	--	-------	--	--	--	-------	-------

w2vst2	(a)	1.165	0.041	28.446	0.000	0.840	0.799
--------	-----	-------	-------	--------	-------	-------	-------

w2vst3	(b)	1.187	0.043	27.693	0.000	0.856	0.764
--------	-----	-------	-------	--------	-------	-------	-------

```
##
## Covariances:
##           Estimate Std.Err z-value P(>|z|) Std.lv Std.all
##   w1comp ~~
##     w2comp      0.335   0.039   8.597   0.000   0.514   0.514
##   .w1vst1 ~~
##     .w2vst1      0.000   0.026   0.018   0.985   0.000   0.001
##   .w1vst2 ~~
##     .w2vst2     -0.033   0.021  -1.517   0.129  -0.033  -0.110
##   .w1vst3 ~~
##     .w2vst3      0.014   0.025   0.539   0.590   0.014   0.033
##
## Intercepts:
##           Estimate Std.Err z-value P(>|z|) Std.lv Std.all
##   .w1vst1      0.000           0.000   0.000
##   .w2vst1      0.000           0.000   0.000
##   .w1vst2 (c) -0.313   0.087  -3.594   0.000  -0.313  -0.272
##   .w1vst3 (d) -0.373   0.092  -4.077   0.000  -0.373  -0.306
##   .w2vst2 (c) -0.313   0.087  -3.594   0.000  -0.313  -0.297
##   .w2vst3 (d) -0.373   0.092  -4.077   0.000  -0.373  -0.333
##   w1comp (h)  2.004   0.036  55.806   0.000   2.219   2.219
##   w2comp (h)  2.004   0.036  55.806   0.000   2.779   2.779
##
## Variances:
##           Estimate Std.Err z-value P(>|z|) Std.lv Std.all
##   .w1vst1      0.471   0.034  13.995   0.000   0.471   0.366
##   .w1vst2      0.218   0.028   7.772   0.000   0.218   0.165
##   .w1vst3      0.338   0.033  10.345   0.000   0.338   0.227
##   .w2vst1      0.576   0.042  13.774   0.000   0.576   0.525
##   .w2vst2      0.400   0.038  10.502   0.000   0.400   0.362
##   .w2vst3      0.522   0.044  11.868   0.000   0.522   0.416
##   w1comp      0.815   0.068  11.956   0.000   1.000   1.000
##   w2comp      0.520   0.046  11.223   0.000   1.000   1.000
```

```
# Fit measures.
fitMeasures(model_ex_2_strong_equal_latent_means_fit, fit.measures = fit_indices)
```

```
##           chisq      df      pvalue      cfi      tli      rmsea rmsea.pvalue
##      14.439    10.000      0.154      0.997      0.996      0.028      0.882
##           srmr
##           0.029
```

```
# Combine the fit measures for the configural and weak model.
fit_measures_all_ex_2 <- rbind(
  fit_measures_all_ex_2,
  strong_equal_latent_means = fitMeasures(model_ex_2_strong_equal_latent_means_fit, fit_indices)
)

# Print fit measures with four decimals.
print(round(fit_measures_all_ex_2, 4))
```

```
##           chisq df pvalue      cfi      tli      rmsea rmsea.pvalue      srmr
## configural      9.9108  5 0.0778 0.9971 0.9914 0.0414      0.5909 0.0181
```

## weak	12.0767	7	0.0981	0.9970	0.9937	0.0355	0.7284	0.0251
## strong	12.3993	9	0.1917	0.9980	0.9967	0.0257	0.8864	0.0253
## strong_equal_latent_means	14.4389	10	0.1539	0.9974	0.9961	0.0278	0.8816	0.0287

The estimate for the mean of the latent variable at both time points is 2.004. We see that our additional constraint on the latent means does not significantly worsen the model fit. Hence, we can say that we have support for the hypothesis of no change in the latent means across time.

- d. Investigate *strict* measurement invariance. Test if the additional constraints of the strict invariance model do not significantly worsen model fit in comparison to the strong invariance model. Report and interpret the results.

For the strict level of measurement invariance, we must do everything we did for the strong model and, on top of that, also constrain the the measurement residuals to be equal across time.

```
# Model syntax.
model_ex_2_strict <- "
  # Measurement part.
  w1comp =~ w1vst1 + a * w1vst2 + b * w1vst3
  w2comp =~ w2vst1 + a * w2vst2 + b * w2vst3

  # Covariance between latent variables.
  w2comp ~~ w1comp

  # Covariances between residuals across time.
  w1vst1 ~~ w2vst1
  w1vst2 ~~ w2vst2
  w1vst3 ~~ w2vst3

  # Fix first intercept at each time point to 0 for identification.
  w1vst1 ~ 0
  w2vst1 ~ 0

  # Constrain the remaining intercepts to be equal across time.
  w1vst2 ~ c * 1
  w1vst3 ~ d * 1
  w2vst2 ~ c * 1
  w2vst3 ~ d * 1

  # Freely estimate means of latent variables.
  w1comp ~ 1
  w2comp ~ 1

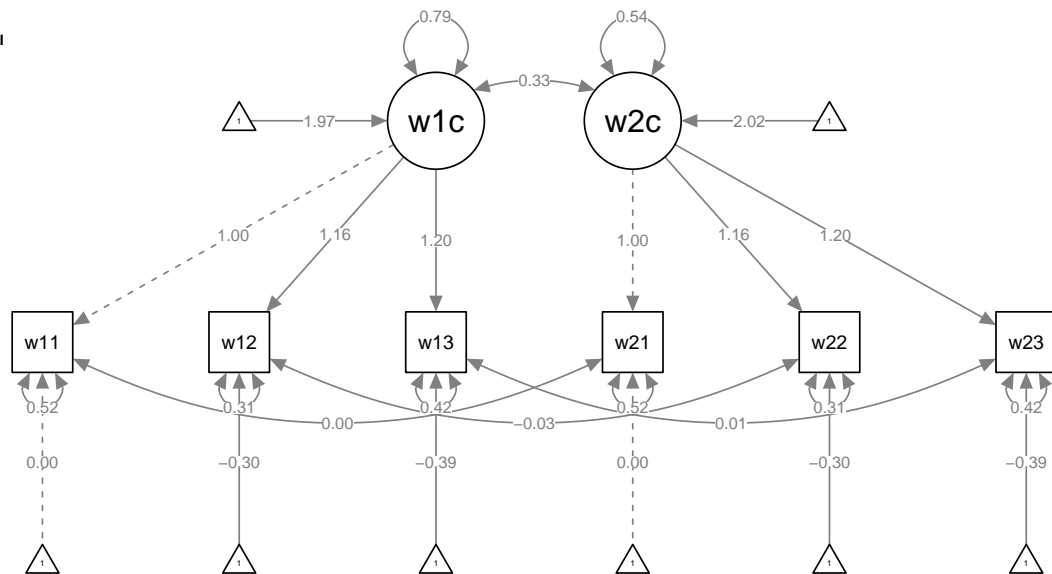
  # Set measurement residuals to be equal across time.
  w1vst1 ~~ e * w1vst1
  w1vst2 ~~ f * w1vst2
  w1vst3 ~~ g * w1vst3
  w2vst1 ~~ e * w2vst1
  w2vst2 ~~ f * w2vst2
  w2vst3 ~~ g * w2vst3
"
```

```
# Fit the model.
model_ex_2_strict_fit <- sem(model_ex_2_strict, data_ex_2)

# Visualize the model.
semPaths(model_ex_2_strict_fit, what = "paths", whatLabels = "est")

# Add a title to the plot.
title("Longitudinal Strict MI", adj = 0)
```

Longitudinal Strict MI



```
# Model summary.
summary(model_ex_2_strict_fit, standardized = TRUE)
```

```
## lavaan 0.6-12 ended normally after 40 iterations
##
## Estimator ML
## Optimization method NLMINB
## Number of model parameters 22
## Number of equality constraints 7
##
## Number of observations 574
##
## Model Test User Model:
##
## Test statistic 60.843
## Degrees of freedom 12
## P-value (Chi-square) 0.000
##
## Parameter Estimates:
##
## Standard errors Standard
## Information Expected
## Information saturated (h1) model Structured
##
## Latent Variables:
```

```

##               Estimate Std.Err z-value P(>|z|) Std.lv Std.all
## w1comp =~
##   w1vst1      1.000
##   w1vst2      (a) 1.162    0.043  27.134    0.000    1.031    0.879
##   w1vst3      (b) 1.197    0.045  26.600    0.000    1.062    0.852
## w2comp =~
##   w2vst1      1.000
##   w2vst2      (a) 1.162    0.043  27.134    0.000    0.855    0.837
##   w2vst3      (b) 1.197    0.045  26.600    0.000    0.881    0.804
##
## Covariances:
##               Estimate Std.Err z-value P(>|z|) Std.lv Std.all
## w1comp ~~
##   w2comp      0.334    0.039    8.552    0.000    0.512    0.512
## .w1vst1 ~~
##   .w2vst1      0.001    0.026    0.029    0.977    0.001    0.001
## .w1vst2 ~~
##   .w2vst2     -0.033    0.022   -1.488    0.137   -0.033   -0.105
## .w1vst3 ~~
##   .w2vst3      0.014    0.026    0.537    0.591    0.014    0.033
##
## Intercepts:
##               Estimate Std.Err z-value P(>|z|) Std.lv Std.all
##   .w1vst1      0.000
##   .w2vst1      0.000
##   .w1vst2      (c) -0.305    0.090   -3.375    0.001   -0.305   -0.260
##   .w1vst3      (d) -0.395    0.095   -4.136    0.000   -0.395   -0.317
##   .w2vst2      (c) -0.305    0.090   -3.375    0.001   -0.305   -0.298
##   .w2vst3      (d) -0.395    0.095   -4.136    0.000   -0.395   -0.360
##   w1comp      1.968    0.044  44.875    0.000    2.218    2.218
##   w2comp      2.023    0.039  52.339    0.000    2.750    2.750
##
## Variances:
##               Estimate Std.Err z-value P(>|z|) Std.lv Std.all
##   .w1vst1      (e) 0.524    0.027  19.086    0.000    0.524    0.400
##   .w1vst2      (f) 0.314    0.025  12.311    0.000    0.314    0.228
##   .w1vst3      (g) 0.424    0.029  14.549    0.000    0.424    0.273
##   .w2vst1      (e) 0.524    0.027  19.086    0.000    0.524    0.492
##   .w2vst2      (f) 0.314    0.025  12.311    0.000    0.314    0.300
##   .w2vst3      (g) 0.424    0.029  14.549    0.000    0.424    0.354
##   w1comp      0.787    0.067  11.737    0.000    1.000    1.000
##   w2comp      0.542    0.048  11.367    0.000    1.000    1.000

```

```

# Fit measures.
fitMeasures(model_ex_2_strict_fit, fit.measures = fit_indices)

```

```

##      chisq      df      pvalue      cfi      tli      rmsea rmsea.pvalue
##      60.843    12.000      0.000      0.972      0.965      0.084      0.003
##      srmr
##      0.034

```

Perform a LRT between the strict model and the strong model.

```

# Perform LRT.
anova(model_ex_2_strict_fit, model_ex_2_strong_fit)

## Chi-Squared Difference Test
##
##              Df      AIC      BIC  Chisq Chisq diff Df diff Pr(>Chisq)
## model_ex_2_strong_fit   9 8859.0 8937.4 12.399
## model_ex_2_strict_fit 12 8901.5 8966.8 60.843      48.444      3 1.713e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

# Combine the fit measures for the configural and weak model.
fit_measures_all_ex_2 <- rbind(
  fit_measures_all_ex_2,
  strict = fitMeasures(model_ex_2_strict_fit, fit_indices)
)

# Print fit measures with four decimals.
print(round(fit_measures_all_ex_2, 4))

```

	chisq	df	pvalue	cfi	tli	rmsea	rmsea.pvalue	srmr
## configural	9.9108	5	0.0778	0.9971	0.9914	0.0414	0.5909	0.0181
## weak	12.0767	7	0.0981	0.9970	0.9937	0.0355	0.7284	0.0251
## strong	12.3993	9	0.1917	0.9980	0.9967	0.0257	0.8864	0.0253
## strong_equal_latent_means	14.4389	10	0.1539	0.9974	0.9961	0.0278	0.8816	0.0287
## strict	60.8430	12	0.0000	0.9716	0.9645	0.0842	0.0035	0.0335

These results indicate that strict measurement invariance is not supported (i.e., the χ^2 is significant and the *CFI* and *RMSEA* worsen in comparison with the strong measurement invariance model).

- e. Consider this statement: “The factor mean invariance test is really a test of the equality of the observed means for the marker variables at each time point.” Is this statement true or false. Explain your answer.

Because factor means are a function of measurement intercepts, tests of factor means are not independent of intercept tests. With the marker variable approach (i.e., where we give the latent variable basically the scale of the marker variable), the mean of the factor is equal to the expected value of the indicator variable.

Part 2. A naïve approach?

We have seen that the R package *lavaan* has full support for multiple group analysis (i.e., as demonstrated in the *lavaan* tutorial and seen in *Exercise 1*). Consider how the data in the dataset *socex1.dat* are currently structured. Would it be possible to re-estimate the models investigated during *Exercise 2* with the multiple group analysis module? Justify your answer.

The data are in wide format. However, even if we transfer the data to long format and add a grouping variable to indicate the measurement occasions, we cannot use the automated procedure for multiple group analysis in *lavaan*. The reason for this is because we have longitudinal data and it makes sense to allow for the error terms to covary across time. Since the automated procedure in *lavaan* fits the model to each group separately (i.e., time point in our case), there is no way to estimate a covariance between the error terms of two separate models.

References

- Beaujean, A. A. (2014). *Latent variable modeling using R: A step by step guide*. Routledge/Taylor & Francis Group.
- Newsom, J. T. (2015). *Longitudinal structural equation modeling: A comprehensive introduction*. Routledge, Taylor and Francis Group.