P.07 - Measurement Invariance

Mihai A. Constantin

November 15, 2022 (12:19:24)

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:

```r
# 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.

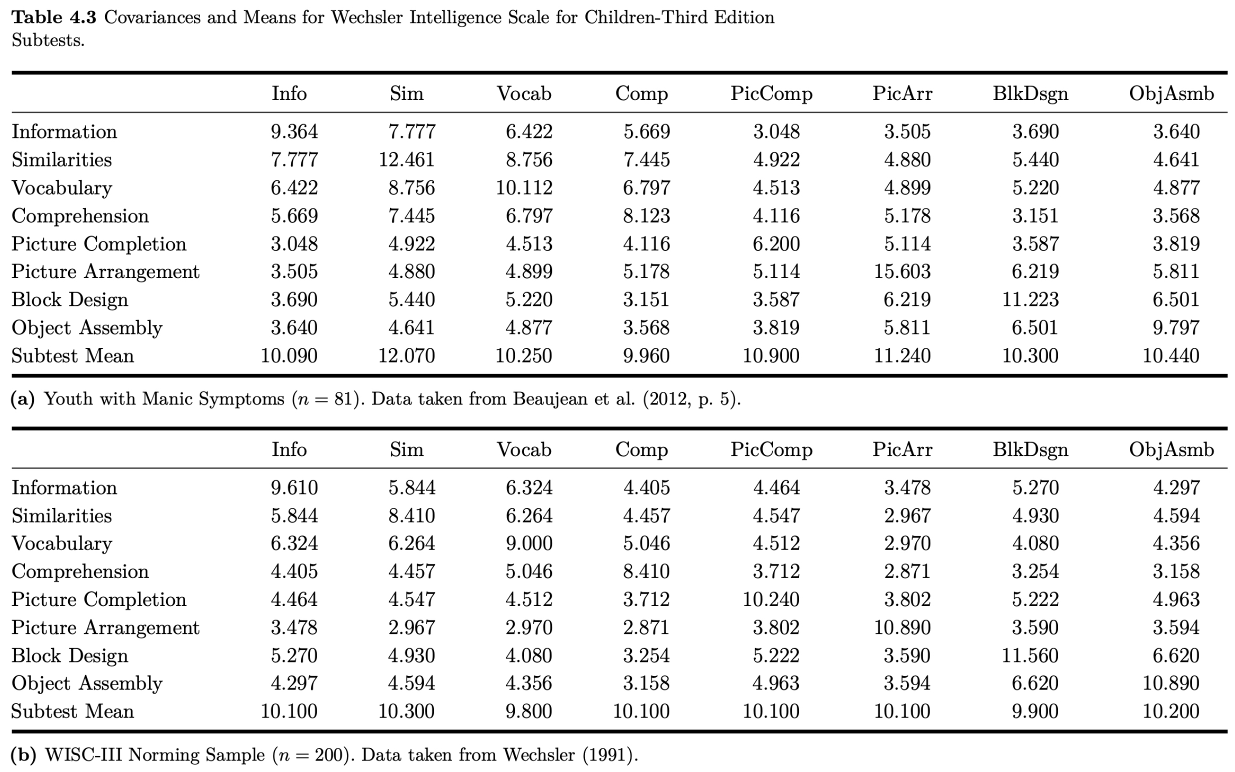


Table 4.3 Covariances and Means for Wechsler Intelligence Scale for Children-Third Edition Subtests.

	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.

```r
# 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).
```

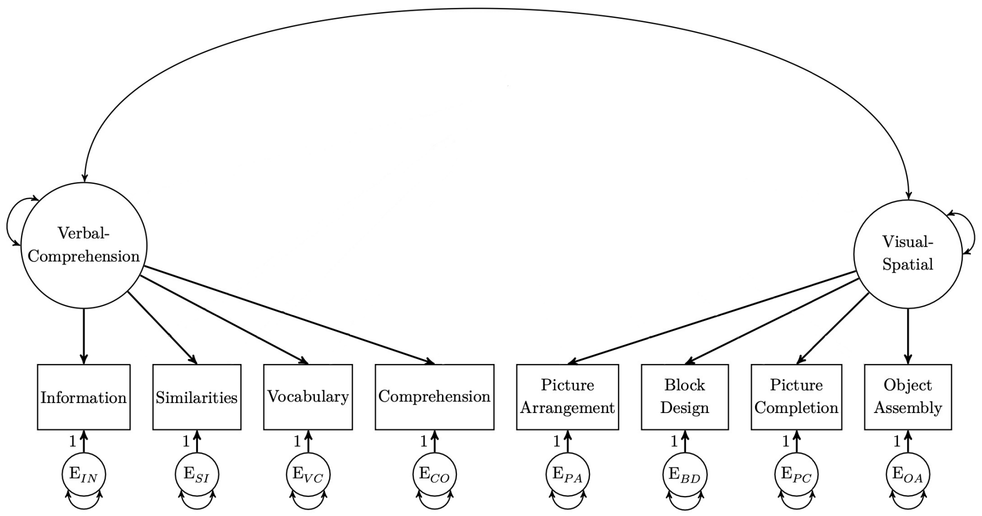


Figure 2: Adaptation of *Figure 4.3* from Beaujean (2014).

```r
# 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.
```

```r
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:

```r
# 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
```

```r
# 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
```

```r
# 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
```

```r
# 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:

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

```r
# 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:

```r
# 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.*

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.

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

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

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

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

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

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

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`.

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

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*.

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

Part 3. Generating some data.

This part is only for the brave.

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 $\boldsymbol{\mu}$ vector of means and the $\boldsymbol{\Sigma}$ 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 for 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`.

```r
# 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
```

```r
# 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
## -0.0001  0.0008  0.0009 -0.0016  0.0000  0.0046  0.0051 -0.0002
```

```
# 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    0.012  0.002  0.003  0.001  0.001 -0.020 -0.009 -0.002
## sim    0.002  0.003  0.005  0.005  0.005 -0.009  0.003 -0.002
## voc    0.003  0.005  0.000  0.003 -0.003 -0.015  0.006 -0.006
## com    0.001  0.005  0.003  0.002  0.005 -0.013 -0.002 -0.008
## p_com  0.001  0.005 -0.003  0.005  0.001 -0.014 -0.003  0.001
## p_arr -0.020 -0.009 -0.015 -0.013 -0.014 -0.076 -0.027 -0.023
## b_des -0.009  0.003  0.006 -0.002 -0.003 -0.027  0.002  0.000
## o_ass -0.002 -0.002 -0.006 -0.008  0.001 -0.023  0.000 -0.011
```

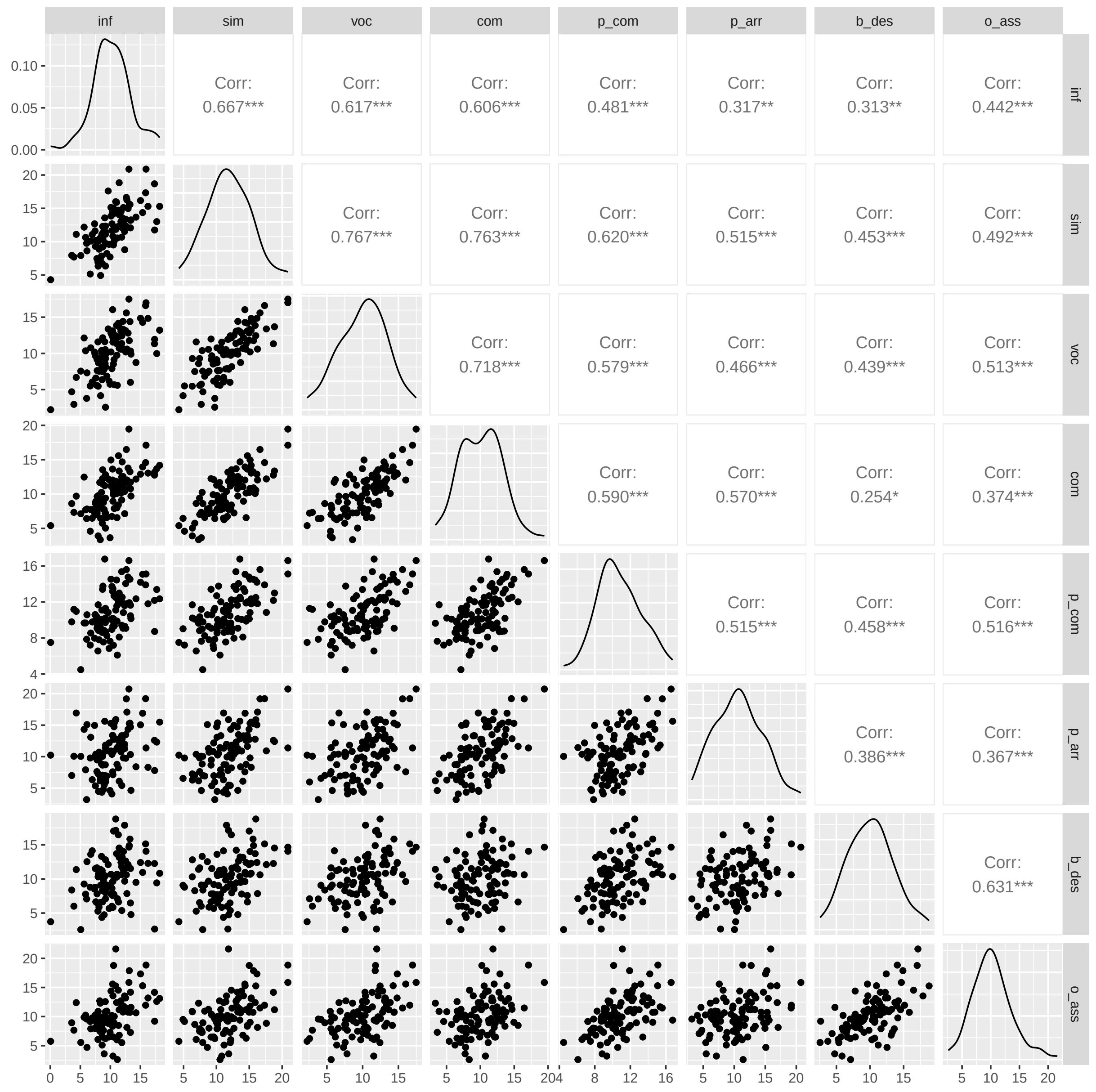
We can see that the differences are quite small. You can test that the more we increase the sample size (i.e., `n`), the smaller the differences become.

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.

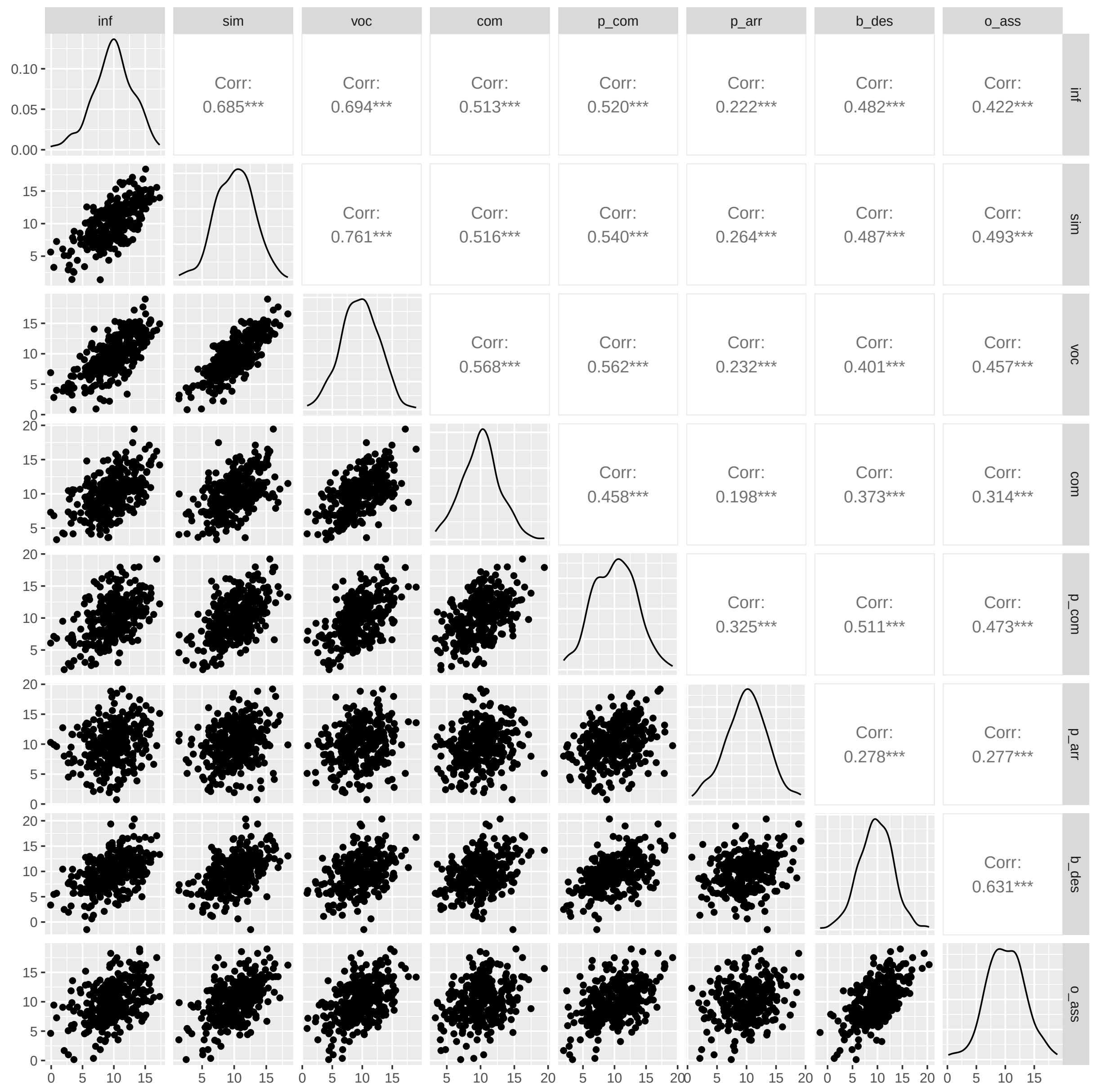
```
# Set seed for the `RNG`.
set.seed(20031993)

# 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.

```
# 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
"

# 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)
```

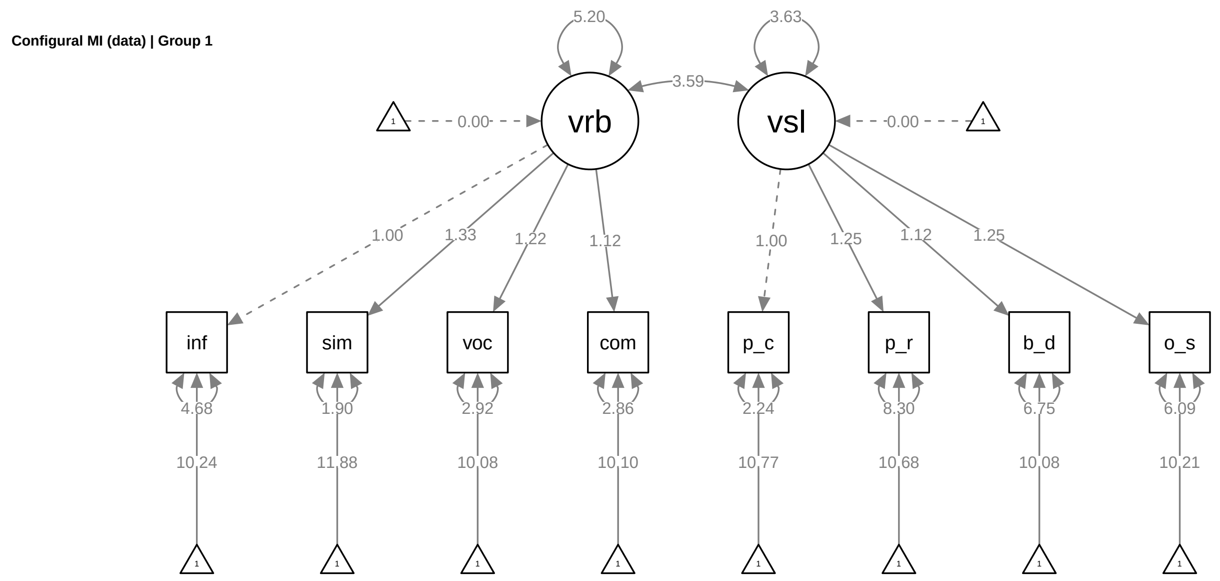
```
## lavaan 0.6-12 ended normally after 101 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of model parameters                        50
##
##   Number of observations per group:
##     manic                                          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                               2.398    0.722
##     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                               0.000    0.000
##     visual            0.000                               0.000    0.000
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .inf               3.621    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)
```

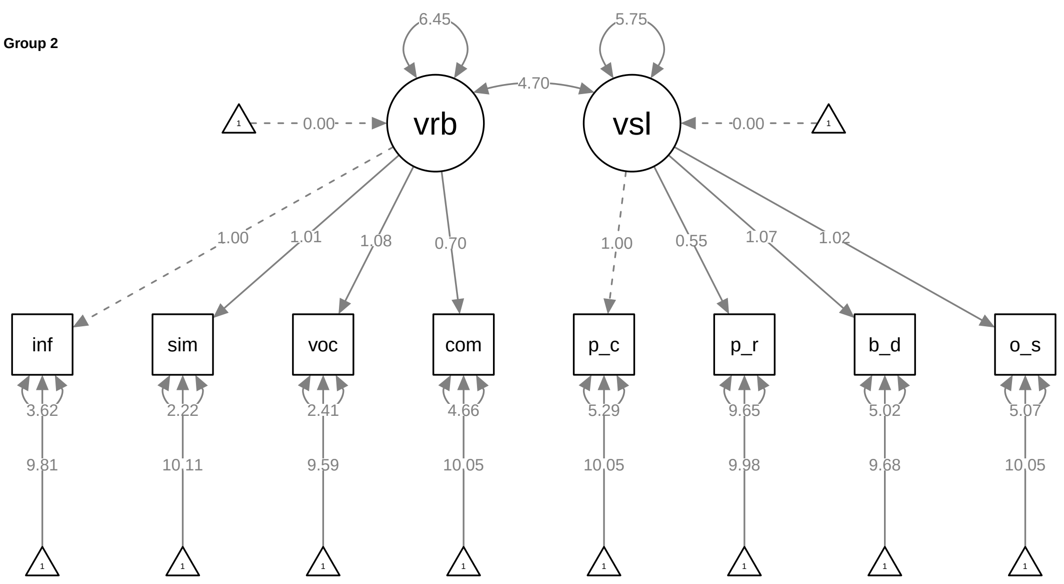
```
##        chisq           df       pvalue          cfi          tli        rmsea rmsea.pvalue
##       97.221       38.000        0.000        0.962        0.943        0.088        0.003
##         srmr
##        0.040
```

```
# 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)
```



```
# 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*.

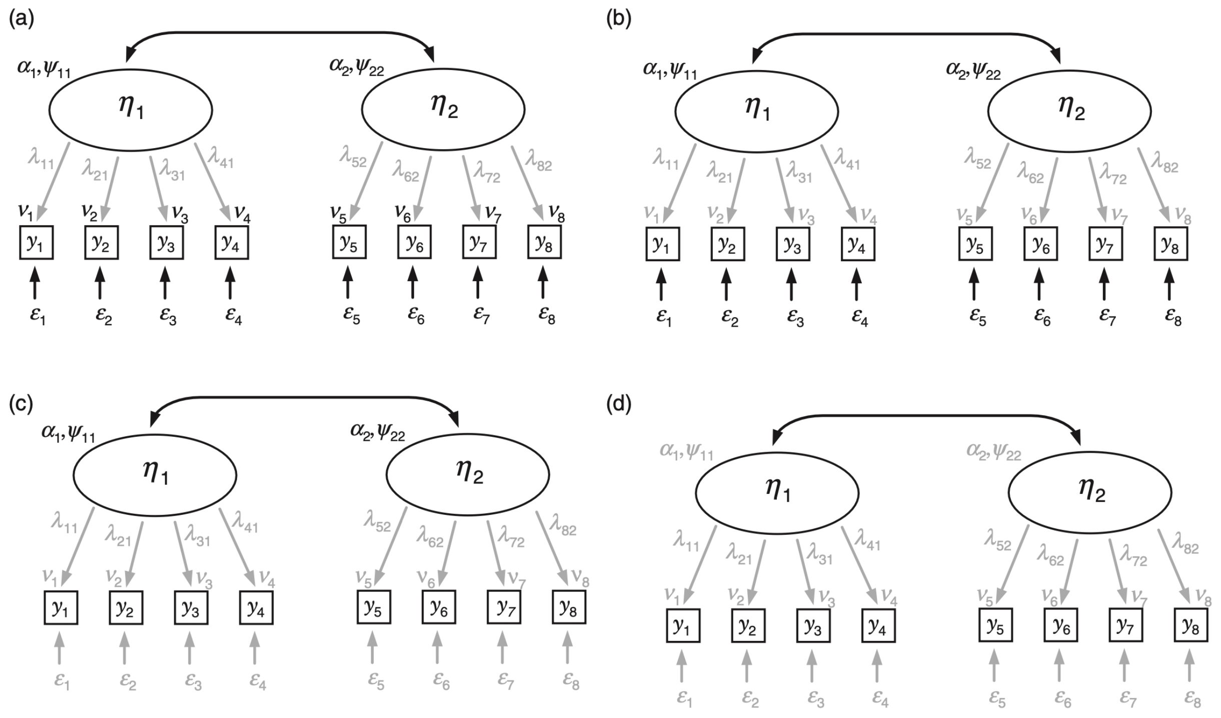

(a)
(b)
(c)
(d)
α_1, ψ_{11}
α_2, ψ_{22}
η_1
η_2
λ_{11} λ_{21} λ_{31} λ_{41} λ_{52} λ_{62} λ_{72} λ_{82}
ν_1 ν_2 ν_3 ν_4 ν_5 ν_6 ν_7 ν_8
y_1 y_2 y_3 y_4 y_5 y_6 y_7 y_8
ε_1 ε_2 ε_3 ε_4 ε_5 ε_6 ε_7 ε_8

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 $\nu_1 = \nu_5$).

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

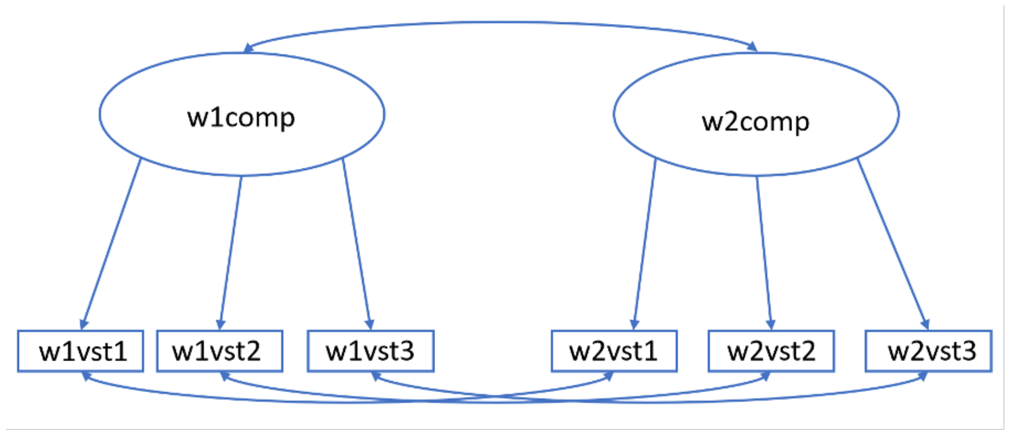


Figure 4: Model example for investigating longitudinal measurement invariance.

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.

```r
# 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.

```r
# 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:

```r
# 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.

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.

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.

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.

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.

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.

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.