# Structural Equation Modeling

P.09 - Statistical Modeling of Panel Data

November 21, 2022 (12:01:23)

---

## Lab Description

For this practical you will need the following packages: `lavaan`, `semPlot`, and `corrplot`. You can install and load these packages using the following code:

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

# Load the packages.
library(lavaan)
library(semPlot)
library(corrplot)
```

Specify which fit measures we are interested in:

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

### Quick Recap

Before we start, let's take a quick look at some of the models discussed during the lecture. They may seem hard, but the two key ideas you should remember are:

1. all these models try to do is describe change across time, so try to identify the auto-regressive or cross-regressive effects
2. each model has its own way of decomposing the variance, e.g., trait vs. state vs. error variance

**The Simplex Model**

- individuals change at a steady rate
- external influences are minimal
- $\hat{r}_{t,t-l} = r^{l}_{t,t-1}$

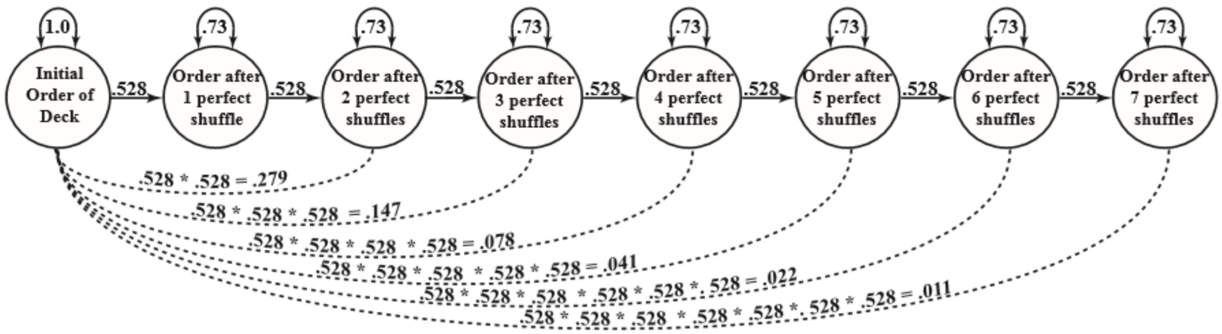



Figure 1: Example of *Simplex* model.

**The Quasi-Simplex Model**

- same as the **Simplex Model**, but it account for measurement error via the estimation of measurement residuals with single-indicator latent variables

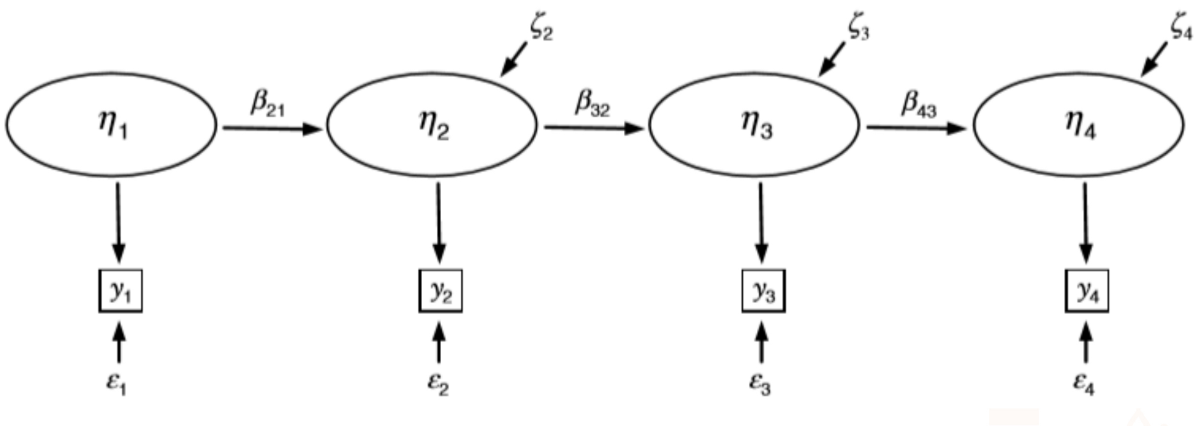



Figure 2: Example of *Quasi-Simplex* model.

### The Univariate STARTS Model

Proposes that at each time point, the measured variable can be a function of three independent latent variables:

- a *stable trait* (i.e., $ST$)
- a *time-varying factor* reflecting an auto-regressive trait (i.e., $ART$)
- a *state factor* reflecting time-specific effects and measurement error (i.e., $S$)

The total variance at one time point is given by $var(ST) + var(ART) + var(S)$.

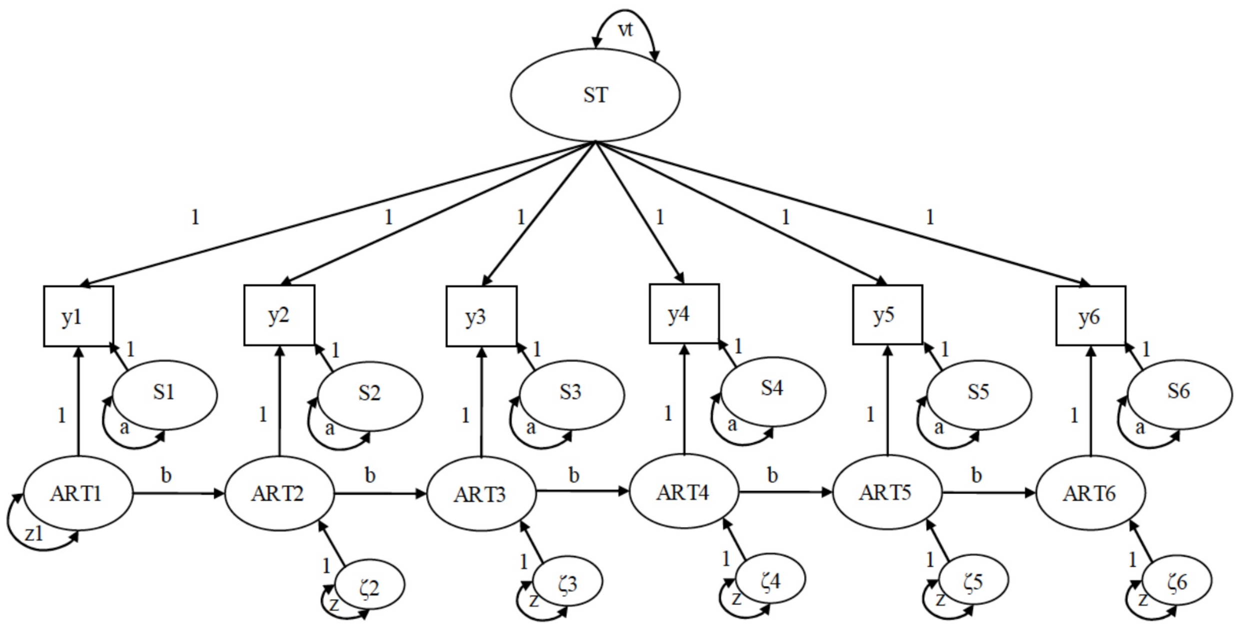



Figure 3: Example of *Univariate STARTS* model.

**The Multivariate STARTS Model**

Same as the **Univariate STARTS Model**, but now the construct is measured by multiple indicators.

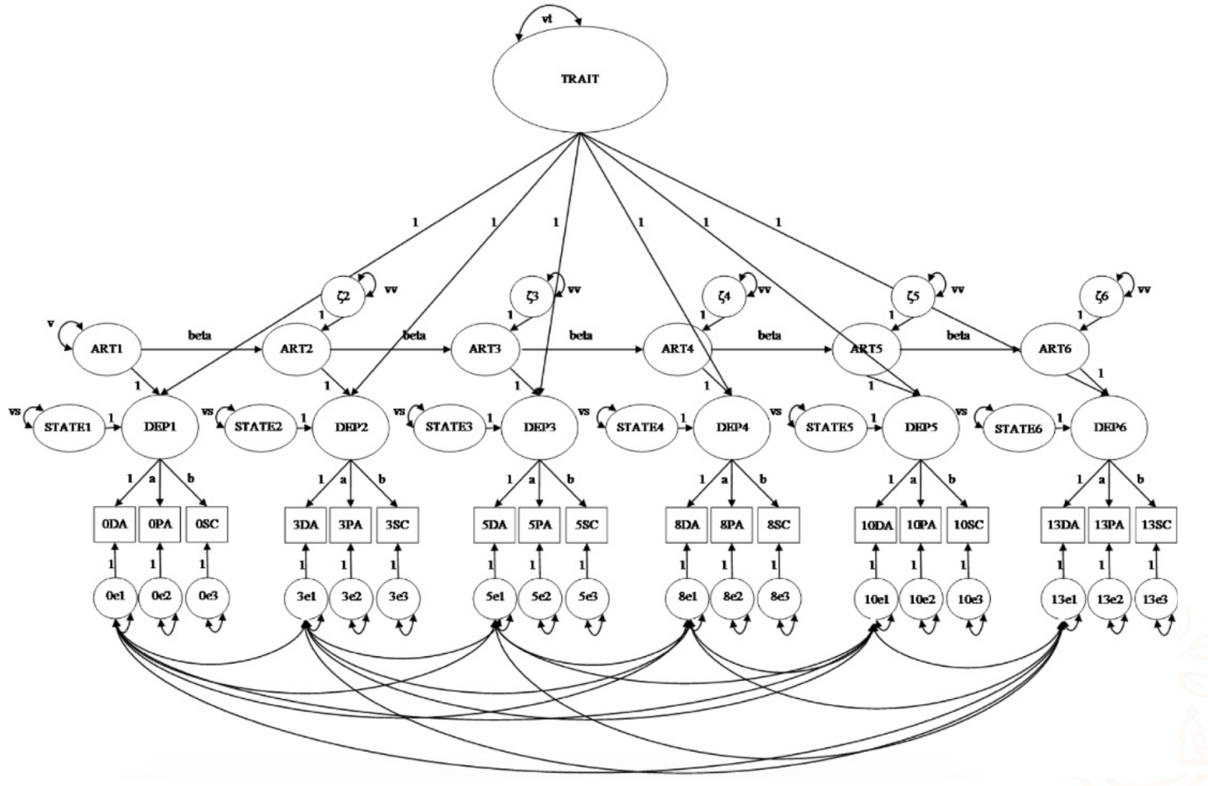



Figure 4: Example of *Multivariate STARTS* model.

## Exercise 1

In this exercise you are going to investigate whether a repeated measurement conforms to the *simplex* or *quasi-simplex* correlation structure. Consider the dataset `health.dat`, which can be found in the folder for the current practical on Canvas. These data are derived from a national health survey with interviews of individuals aged 50 years and above conducted biannually. You are going to analyze the self-rated health question about overall health collected over six waves. Ratings from this question were from 1 (*poor*) to 5 (*excellent*). The repeated measurement variables are: `srh1`, `srh2`, `srh3`, `srh4`, `srh5`, `srh6`. To get you started, you can use the following code to load the data and set the variable names.

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_1 <- read.table("health.dat")

# Inspect the data.
View(data_ex_1)
```

Set the variable names.

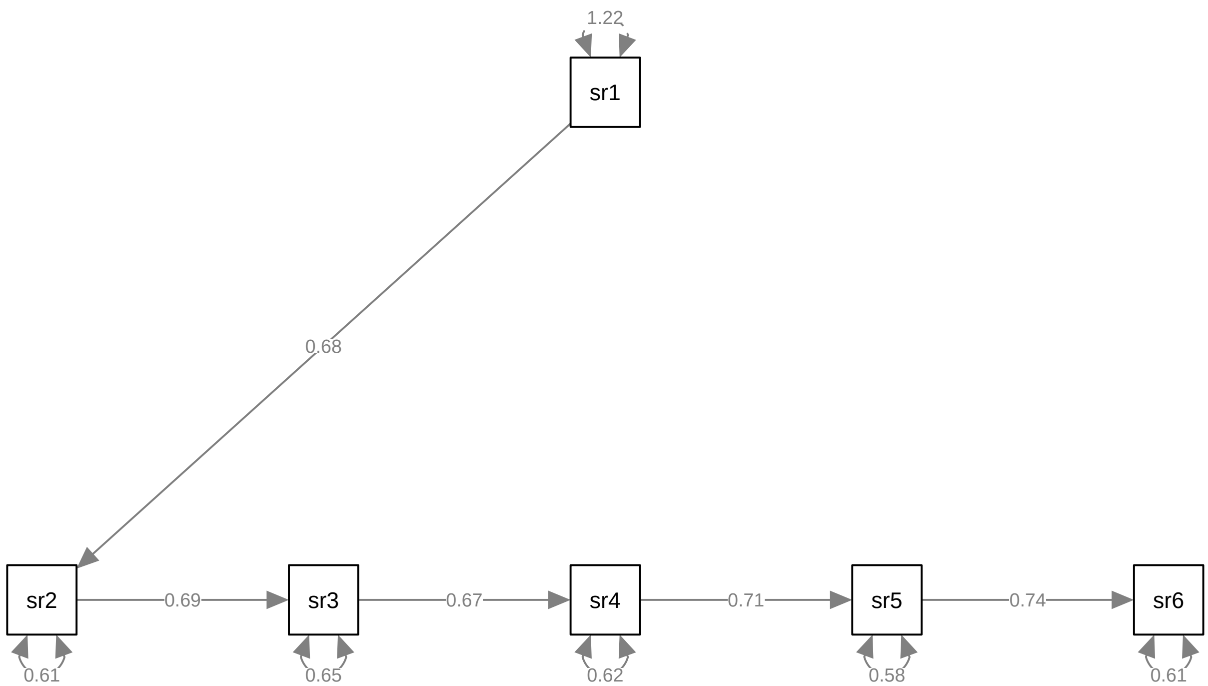
```r
# Variable names.
variable_ex_1_names = c(
    "age", "srh1", "srh2", "srh3", "srh4", "srh5", "srh6", "bmi1",
    "bmi2", "bmi3", "bmi4", "bmi5", "bmi6", "cesdna1", "cesdpa1", "cesdso1",
    "cesdna2", "cesdpa2", "cesdso2", "cesdna3", "cesdpa3", "cesdso3",
    "cesdna4", "cesdpa4", "cesdso4", "cesdna5", "cesdpa5", "cesdso5",
    "cesdna6", "cesdpa6", "cesdso6", "diab1", "diab2", "diab3 ", "diab4", "diab5", "diab6"
)

# Set the names.
names(data_ex_1) <- variable_ex_1_names

# List variables.
str(data_ex_1)
```

```
## 'data.frame':    5335 obs. of  37 variables:
##  $ age    : num  55.1 63.3 58.6 62.3 59.7 ...
##  $ srh1   : num  3.22 2.94 2.06 3.1 5.04 ...
##  $ srh2   : num  2.845 0.475 2.981 5.187 4.843 ...
##  $ srh3   : num  1.581 0.666 3.397 3.966 3.423 ...
##  $ srh4   : num  1.25 3.08 2.93 3.11 4.04 ...
##  $ srh5   : num  2.11 3.31 2.91 3.39 3.68 ...
##  $ srh6   : num  0.717 3.427 2.211 4.089 2.891 ...
##  $ bmi1   : num  27.1 24.7 13.9 23.7 23.8 ...
##  $ bmi2   : num  29.4 27.1 12.5 24.7 25.6 ...
##  $ bmi3   : num  29.7 27.9 13.4 24.2 25.4 ...
##  $ bmi4   : num  29.2 25.3 15.8 26.1 25.3 ...
##  $ bmi5   : num  27.9 26.7 15.9 25.4 28.2 ...
##  $ bmi6   : num  27.8 28 15.4 24.8 31.2 ...
```

```
##  $ cesdna1: num  0.743 0.1649 1.8722 1.3137 0.0949 ...
##  $ cesdpa1: num  0.4369 -0.0335 1.4751 0.4012 -0.8426 ...
##  $ cesdso1: num  1.01 1.55 2.19 1.51 1.03 ...
##  $ cesdna2: num  0.128 1.341 1.135 0.235 -0.114 ...
##  $ cesdpa2: num  0.0388 0.3245 0.5635 0.8914 -0.2595 ...
##  $ cesdso2: num  0.566 1.446 0.898 1.029 0.121 ...
##  $ cesdna3: num  1.282 0.646 1.654 1.12 -0.337 ...
##  $ cesdpa3: num  0.0045 0.1149 0.9773 0.0697 -0.5989 ...
##  $ cesdso3: num  0.982 1.147 0.544 1.091 -0.223 ...
##  $ cesdna4: num  0.766 0.452 0.947 0.9 -0.21 ...
##  $ cesdpa4: num  1.068 0.43 1.138 -0.281 -0.22 ...
##  $ cesdso4: num  1.345 1.921 0.78 1.28 0.534 ...
##  $ cesdna5: num  1.663 0.486 1.04 1.849 -0.978 ...
##  $ cesdpa5: num  0.4749 0.0759 0.854 0.3355 -0.3516 ...
##  $ cesdso5: num  1.149 1.395 1.149 0.909 0.309 ...
##  $ cesdna6: num  0.775 -0.657 1.105 1.009 0.291 ...
##  $ cesdpa6: num  0.929 -0.581 0.821 0.586 0.857 ...
##  $ cesdso6: num  0.522 1.077 1.341 1.256 0.957 ...
##  $ diab1  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ diab2  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ diab3  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ diab4  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ diab5  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ diab6  : int  0 0 0 0 0 0 0 0 0 0 ...
```

a. Estimate the perfect simplex model for these six repeated measurement. Request a standardized solution, and evaluate the fit of this model.

```
# Model syntax.
model_ex_1_simplex <- "
    srh2 ~ srh1
    srh3 ~ srh2
    srh4 ~ srh3
    srh5 ~ srh4
    srh6 ~ srh5
"

# Fit model.
model_ex_1_simplex_fit <- sem(model_ex_1_simplex, data = data_ex_1)

# Visualize the model.
semPaths(model_ex_1_simplex_fit, what = "paths", whatLabels = "est")
```




```r
# Model summary.
summary(model_ex_1_simplex_fit, standardized = TRUE)
```

```
## lavaan 0.6-12 ended normally after 1 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of model parameters                        10
##
##   Number of observations                          5335
##
## Model Test User Model:
##
##   Test statistic                              3564.560
##   Degrees of freedom                                10
##   P-value (Chi-square)                           0.000
##
## Parameter Estimates:
##
##   Standard errors                             Standard
##   Information                                 Expected
##   Information saturated (h1) model          Structured
##
## Regressions:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   srh2 ~
##     srh1              0.679    0.010   69.903    0.000    0.679    0.691
##   srh3 ~
##     srh2              0.686    0.010   67.190    0.000    0.686    0.677
##   srh4 ~
##     srh3              0.669    0.010   68.353    0.000    0.669    0.683
```

```
##   srh5 ~
##     srh4              0.708    0.010   72.953    0.000    0.708    0.707
##   srh6 ~
##     srh5              0.738    0.010   74.163    0.000    0.738    0.712
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .srh2              0.612    0.012   51.648    0.000    0.612    0.522
##    .srh3              0.653    0.013   51.648    0.000    0.653    0.542
##    .srh4              0.616    0.012   51.648    0.000    0.616    0.533
##    .srh5              0.580    0.011   51.648    0.000    0.580    0.501
##    .srh6              0.612    0.012   51.648    0.000    0.612    0.492
```

```
# Fit measures.
fitMeasures(model_ex_1_simplex_fit, fit.measures = fit_indices)
```

```
##       chisq          df      pvalue         cfi         tli       rmsea rmsea.pvalue
##    3564.560      10.000       0.000       0.832       0.748       0.258        0.000
##        srmr
##       0.202
```

We know that for a **simplex** model, the individuals are changing at a steady rate and external influences affecting the rate of change are minimal. We see that this is not the case with the model we just fit. The observed correlations do not decease exponentially with additional lag lengths. The model has poor fit to the data.

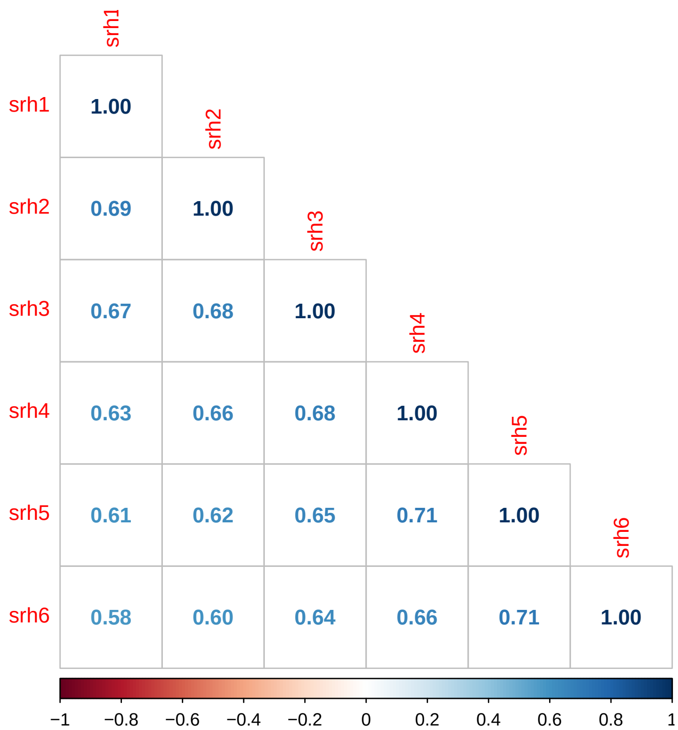
Note that you can also fit again the model in `model_ex_1_simplex`, but this time add equality constraints for the auto-regressive paths. Then, compare the fit of this *perfect* **simplex** model to `model_ex_1_simplex`. That way you can tell more precisely whether your data does indeed conform to a perfect simplex structure.

b. Inspect the correlation of the first time point measurement ($t_1$) with later time points, and the standardized auto-regression coefficients. Does the pattern of these coefficients provide evidence that the perfect simplex model holds?

```
# Subset data.
data_ex_1_subset <- data_ex_1[, c("srh1", "srh2", "srh3", "srh4", "srh5", "srh6")]

# Compute correlations.
ex_1_cors_simplex <- cor(data_ex_1_subset)

# Print the correlations.
round(ex_1_cors_simplex, 3)
```

```
##       srh1  srh2  srh3  srh4  srh5  srh6
## srh1 1.000 0.691 0.669 0.626 0.607 0.585
## srh2 0.691 1.000 0.677 0.656 0.624 0.601
## srh3 0.669 0.677 1.000 0.683 0.654 0.637
## srh4 0.626 0.656 0.683 1.000 0.707 0.656
## srh5 0.607 0.624 0.654 0.707 1.000 0.712
## srh6 0.585 0.601 0.637 0.656 0.712 1.000
```

```
# Visualize the correlations.
corrplot(ex_1_cors_simplex, method = "number", type = "lower")
```



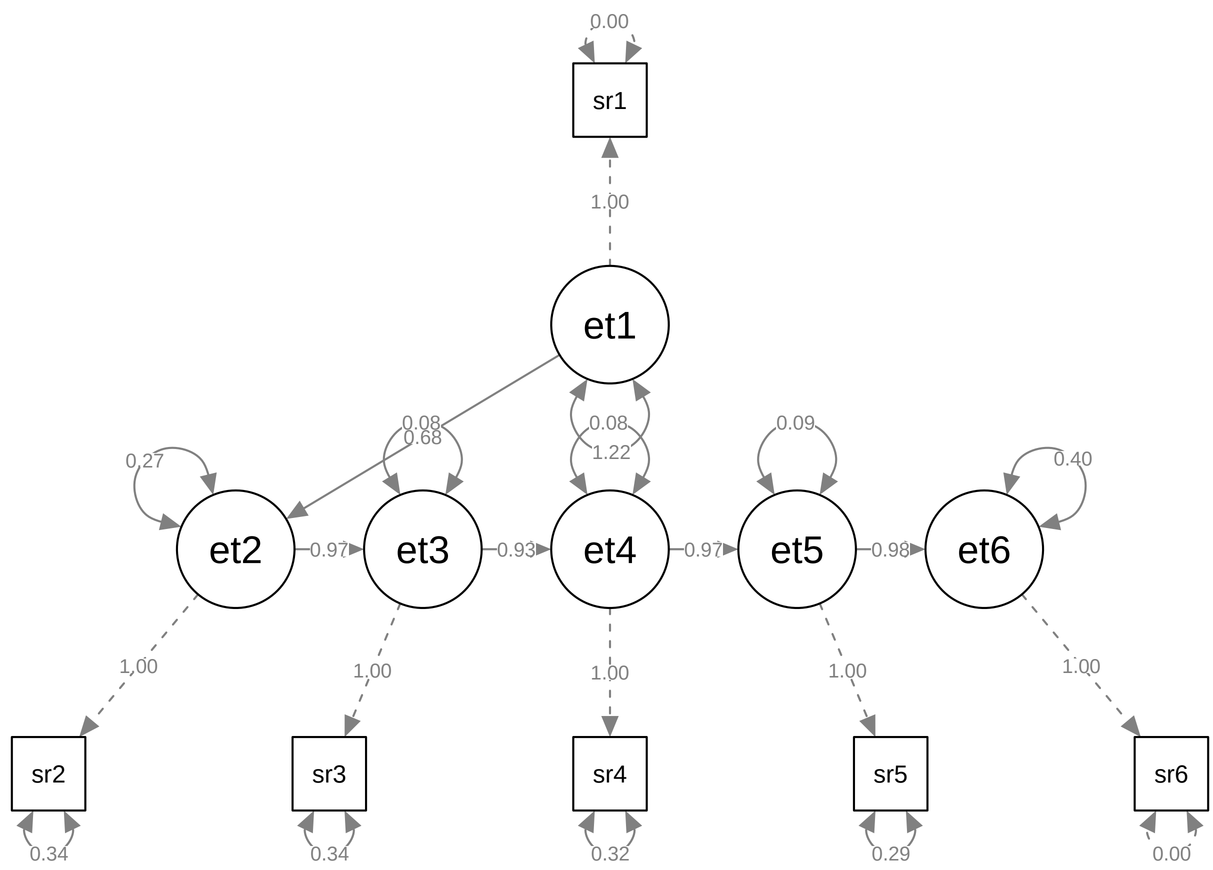

Inspection of correlation of the $t_1$ with later time points (i.e., $r_{12} = .619$, $r_{13} = .669$, $r_{14} = .626$, $r_{15} = .607$, $r_{16} = .585$) shows that correlations decline at greater lag length, but not at an exponential rate. Also, standardized auto-regression coefficients i.e., (.691, .677, .683, .707, .712) show increasing values due to cumulative increases in stability estimates.

c. Now estimate the quasi-simplex model. For identification purposes, set the measurement residual variances of the first and last measurement to 0. Again, request a standardized solution and evaluate the fit of this model.

**Note.** The first and last residuals must be set to 0 for identification. Parameter estimates involving first and last variable do not take into account measurement error.

```
# Model syntax.
model_ex_1_quasi_simplex <- "
    # Measurement part.
    eta1 =~ 1 * srh1
    eta2 =~ 1 * srh2
    eta3 =~ 1 * srh3
    eta4 =~ 1 * srh4
    eta5 =~ 1 * srh5
    eta6 =~ 1 * srh6

    # Set residuals to 0 for identification.
    srh1 ~~ 0 * srh1
    srh6 ~~ 0 * srh6
```

```
    # Freely estimate the remaining residuals.
    srh2 ~~ srh2
    srh3 ~~ srh3
    srh4 ~~ srh4
    srh5 ~~ srh5

    # Structural part.
    eta2 ~ eta1
    eta3 ~ eta2
    eta4 ~ eta3
    eta5 ~ eta4
    eta6 ~ eta5
"

# Fit model.
model_ex_1_quasi_simplex_fit <- sem(model_ex_1_quasi_simplex, data = data_ex_1)

# Visualize the model.
semPaths(model_ex_1_quasi_simplex_fit, what = "paths", whatLabels = "est")
```




```
# Model summary.
summary(model_ex_1_quasi_simplex_fit, standardized = TRUE)
```

```
## lavaan 0.6-12 ended normally after 31 iterations
##
##   Estimator                                         ML
```

```
##   Optimization method                           NLMINB
##   Number of model parameters                        15
##
##   Number of observations                          5335
##
## Model Test User Model:
##
##   Test statistic                                23.206
##   Degrees of freedom                                 6
##   P-value (Chi-square)                           0.001
##
## 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
##   eta1 =~
##     srh1              1.000                               1.103    1.000
##   eta2 =~
##     srh2              1.000                               0.912    0.842
##   eta3 =~
##     srh3              1.000                               0.930    0.847
##   eta4 =~
##     srh4              1.000                               0.915    0.851
##   eta5 =~
##     srh5              1.000                               0.932    0.866
##   eta6 =~
##     srh6              1.000                               1.115    1.000
##
## Regressions:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   eta2 ~
##     eta1              0.679    0.010   69.903    0.000    0.821    0.821
##   eta3 ~
##     eta2              0.973    0.015   63.365    0.000    0.955    0.955
##   eta4 ~
##     eta3              0.933    0.014   66.466    0.000    0.948    0.948
##   eta5 ~
##     eta4              0.967    0.014   69.428    0.000    0.949    0.949
##   eta6 ~
##     eta5              0.984    0.014   71.137    0.000    0.823    0.823
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .srh1              0.000                               0.000    0.000
##    .srh6              0.000                               0.000    0.000
##    .srh2              0.341    0.010   33.659    0.000    0.341    0.290
##    .srh3              0.341    0.010   35.184    0.000    0.341    0.283
```

```
##    .srh4              0.318    0.009   34.802    0.000    0.318    0.275
##    .srh5              0.289    0.009   30.787    0.000    0.289    0.250
##     eta1              1.217    0.024   51.648    0.000    1.000    1.000
##    .eta2              0.272    0.009   29.195    0.000    0.326    0.326
##    .eta3              0.076    0.009    8.428    0.000    0.088    0.088
##    .eta4              0.084    0.008   10.230    0.000    0.100    0.100
##    .eta5              0.086    0.009    9.630    0.000    0.099    0.099
##    .eta6              0.402    0.011   37.651    0.000    0.323    0.323
```

```r
# Fit measures.
fitMeasures(model_ex_1_quasi_simplex_fit, fit.measures = fit_indices)
```

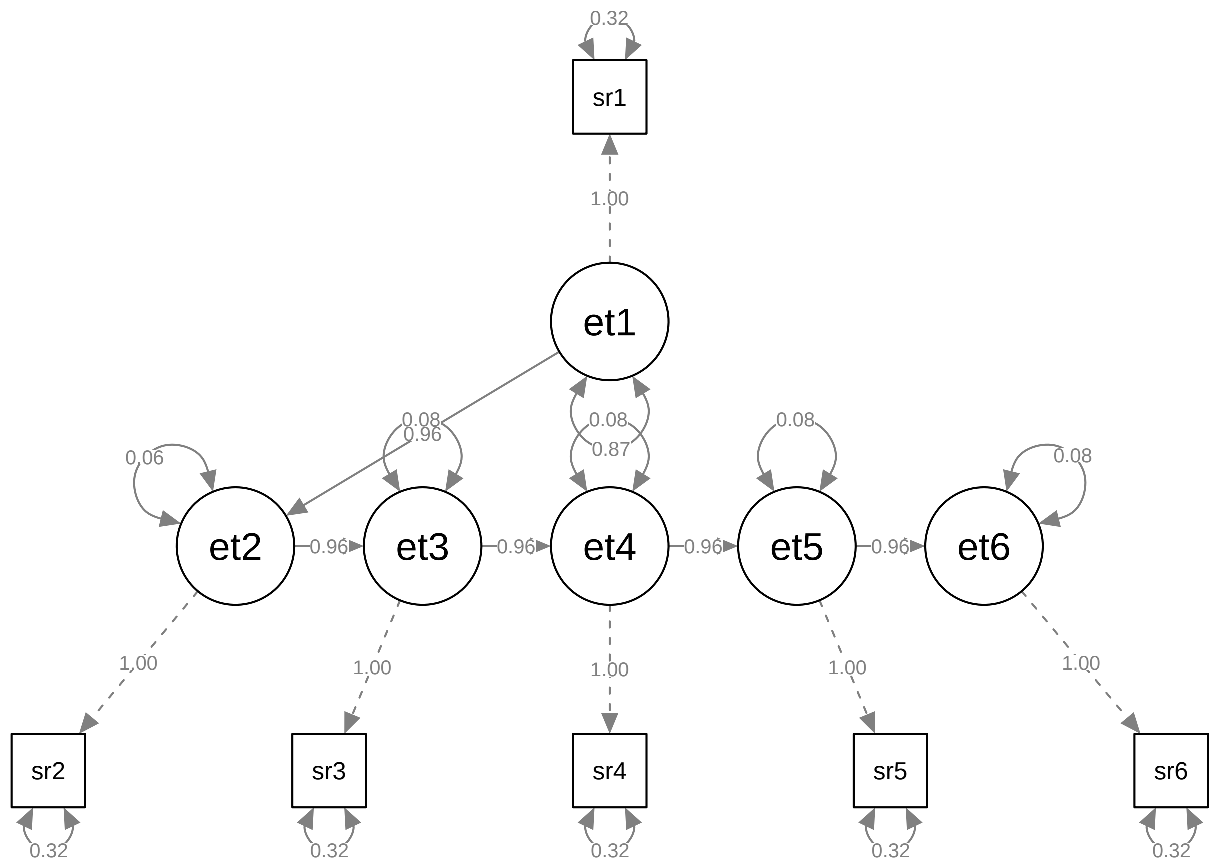
```
##        chisq           df       pvalue          cfi          tli        rmsea rmsea.pvalue
##       23.206        6.000        0.001        0.999        0.998        0.023        1.000
##         srmr
##        0.005
```

We see that the fit of this model is better.

An alternative parametrization, more closely related to what was discussed during the lecture, is to (1) constrain all residual variances to be equal across time, and also (2) constrain the auto-regressive coefficients to be equal.

```r
# Model syntax.
model_ex_1_quasi_simplex_alt <- "
    # Measurement part.
    eta1 =~ 1 * srh1
    eta2 =~ 1 * srh2
    eta3 =~ 1 * srh3
    eta4 =~ 1 * srh4
    eta5 =~ 1 * srh5
    eta6 =~ 1 * srh6

    # Freely estimate the remaining residuals.
    srh1 ~~ b * srh1
    srh2 ~~ b * srh2
    srh3 ~~ b * srh3
    srh4 ~~ b * srh4
    srh5 ~~ b * srh5
    srh6 ~~ b * srh6

    # Structural part.
    eta2 ~ a * eta1
    eta3 ~ a * eta2
    eta4 ~ a * eta3
    eta5 ~ a * eta4
    eta6 ~ a * eta5
"

# Fit model.
model_ex_1_quasi_simplex_alt_fit <- sem(model_ex_1_quasi_simplex_alt, data = data_ex_1)

# Visualize the model.
```

```r
semPaths(model_ex_1_quasi_simplex_alt_fit, what = "paths", whatLabels = "est")
```




And we can indeed see that the new model `model_ex_1_quasi_simplex_alt` has a similar fit to `model_ex_1_quasi_simplex`.

```r
# Print the fit indices side-by-side.
knitr::kable(round(cbind(
    model_ex_1_quasi_simplex = fitMeasures(model_ex_1_quasi_simplex_fit, fit.measures = fit_indices),
    model_ex_1_quasi_simplex_alt = fitMeasures(model_ex_1_quasi_simplex_alt_fit, fit.measures = fit_indices)
), 4))
```

| | model_ex_1_quasi_simplex | model_ex_1_quasi_simplex_alt |
|---|---:|---:|
| chisq | 23.2060 | 62.3873 |
| df | 6.0000 | 13.0000 |
| pvalue | 0.0007 | 0.0000 |
| cfi | 0.9992 | 0.9977 |
| tli | 0.9980 | 0.9973 |
| rmsea | 0.0232 | 0.0267 |
| rmsea.pvalue | 1.0000 | 1.0000 |
| srmr | 0.0047 | 0.0185 |

d. Perform a Likelihood Ratio Test (LRT) of the perfect simplex model against the quasi-simplex model. Interpret the result of this test.

Perform a LRT via the `anova` function in `R`.

```r
# LRT.
anova(model_ex_1_simplex_fit, model_ex_1_quasi_simplex_fit)
```

```
## Chi-Squared Difference Test
##
##                              Df   AIC   BIC    Chisq Chisq diff Df diff Pr(>Chisq)
## model_ex_1_quasi_simplex_fit  6 75370 75469   23.206
## model_ex_1_simplex_fit       10 62712 62778 3564.560     3541.4       4  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Also compare the fit indices

```r
# Store fit measures.
ex_1_fit_measures <- rbind(
    simplex = fitMeasures(model_ex_1_simplex_fit, fit.measures = fit_indices),
    quasi_simplex = fitMeasures(model_ex_1_quasi_simplex_fit, fit.measures = fit_indices)
)

# Print fit measures.
round(ex_1_fit_measures, 3)
```

```
##                  chisq df pvalue   cfi   tli rmsea rmsea.pvalue  srmr
## simplex       3564.560 10  0.000 0.832 0.748 0.258            0 0.202
## quasi_simplex   23.206  6  0.001 0.999 0.998 0.023            1 0.005
```

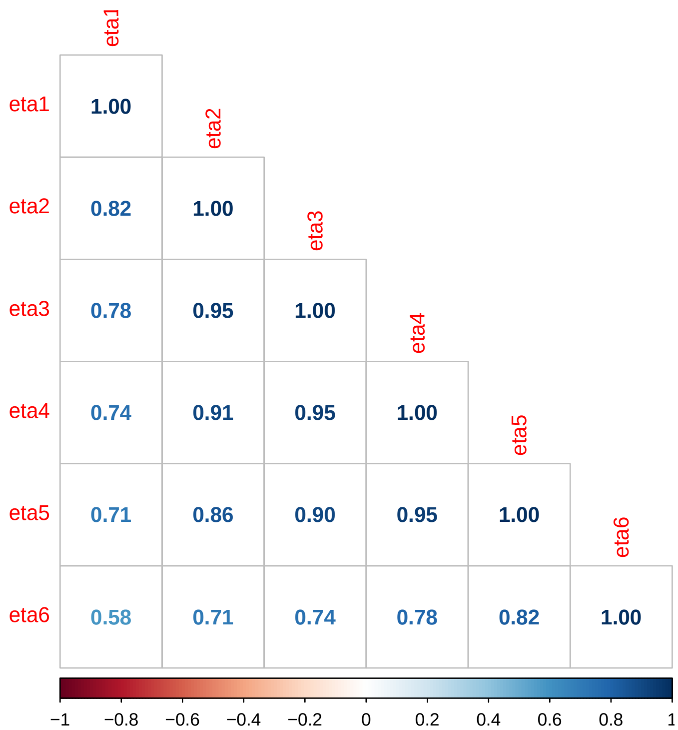
The model with measurement error (i.e., the **quasi-simplex** model) fits significantly better than **simplex** model.

e. Obtain the estimated correlations among the latent variables, using the `lavInspect()` function. Note that you can check the documentation for this function by running `?lavInspect` in `R`.

```r
# Obtain correlations.
ex_1_cors_quasi_simplex <- lavInspect(model_ex_1_quasi_simplex_fit, "cor.lv")

# Print the correlations.
round(ex_1_cors_quasi_simplex, 3)
```

```
##       eta1  eta2  eta3  eta4  eta5  eta6
## eta1 1.000
## eta2 0.821 1.000
## eta3 0.784 0.955 1.000
## eta4 0.743 0.906 0.948 1.000
## eta5 0.706 0.860 0.900 0.949 1.000
## eta6 0.581 0.707 0.741 0.781 0.823 1.000
```

```r
# Visualize the correlations.
corrplot(ex_1_cors_quasi_simplex, method = "number", type = "lower")
```




f. Inspect the correlation of the second time point ($t_2$) measurement with later time points, and the standardized auto-regression coefficients. Does the pattern of these coefficients provide evidence that the data conform to the simplex structure, if measurement error is taken into account for the measurement at $t_2$ to $t_5$?

The estimated correlations suggest conformity to simplex correlation structure. For example, if we look at latent factor correlations between $t_2$ and $t_4$ (i.e., $r = .906$), and $t_2$ and $t_5$ (i.e., $r = .860$), these values are close to what we would obtain if the latent correlation between $t_2$ and $t_3$ (i.e., $r = .955$) is raised to the second and third power, respectively: $.955^2 = .912$, and $.955^3 = .871$. Also note that, once measurement error is accounted for, we see little evidence of cumulative increase in stability coefficients across the middle three auto-regression coefficients.

### Exercise 2

Consider the *Cross-Lagged Panel Model* depicted in *Figure 5*.

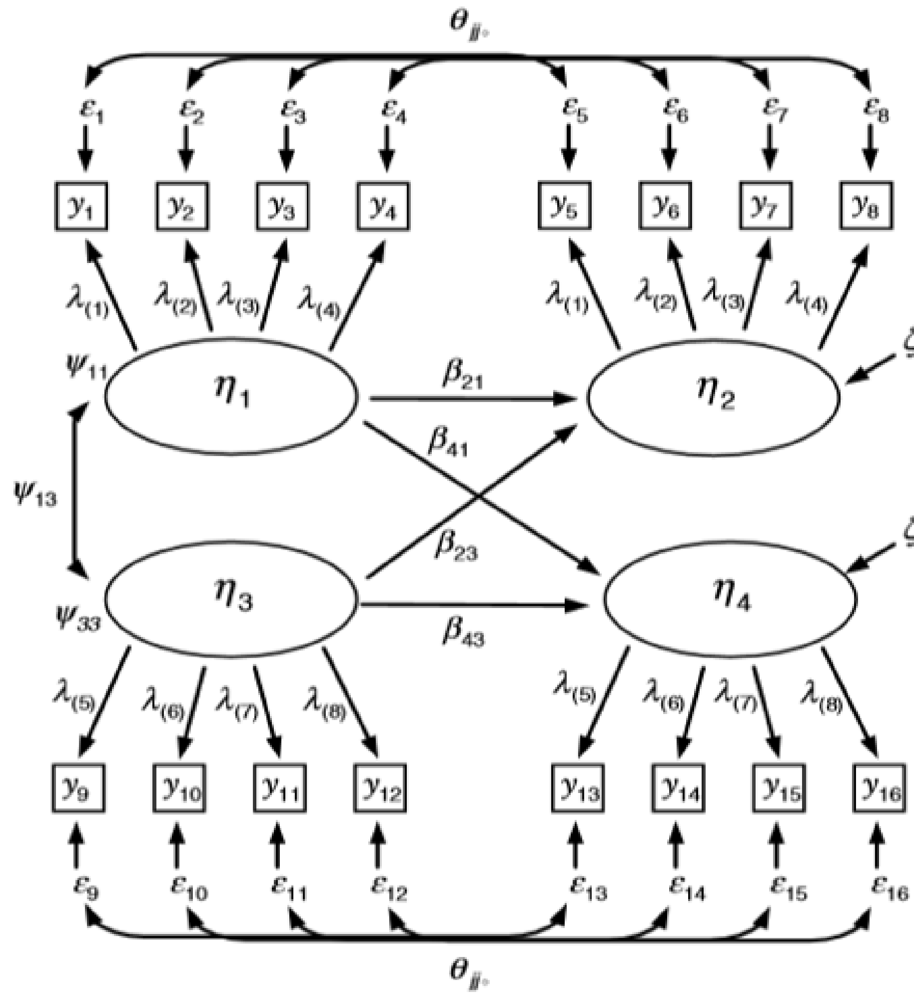



Figure 5: Example of *Cross-Lagged Panel Model* with four latent variables.

Using the `socex1.dat` data, estimate a similar model to investigate the bidirectional effects between *positive affect* (PA) and *unwanted advice* (UA). The specifications of the model are as follows:

- `w1posaff` (latent variable *PA* at $t_1$) with indicators: `w1happy`, `w1enjoy`, `w1satis`, `w1joyful`, and `w1please`
- `w2posaff` (latent variable *PA* at $t_2$) with indicators: `w2happy`, `w2enjoy`, `w2satis`, `w2joyful`, and `w2please`
- `w1unw` (latent variable *UA* at $t_1$) with indicators: `w1unw1`, `w1unw2`, and `w1unw3`
- `w2unw` (latent variable *UA* at $t_2$) with indicators: `w2unw1`, `w2unw2`, and `w2unw3`

In addition, in this model you should:

- Do not use the marker method to set the scale of the latent variables, but free the marker variables and set exogenous factor variance for first occasion latent variables for identification purpose to 1.

- Impose equality constraints for the loadings between the different measurement moments for repeated indicators.
- Estimate the cross-lagged and auto-regressive effects.
- Include correlated measurement residuals.

To get you started, you can use the following `R` code to load the data and set the variable names.

Load the data.

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

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

Create and 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 ...
```

Answer the following:

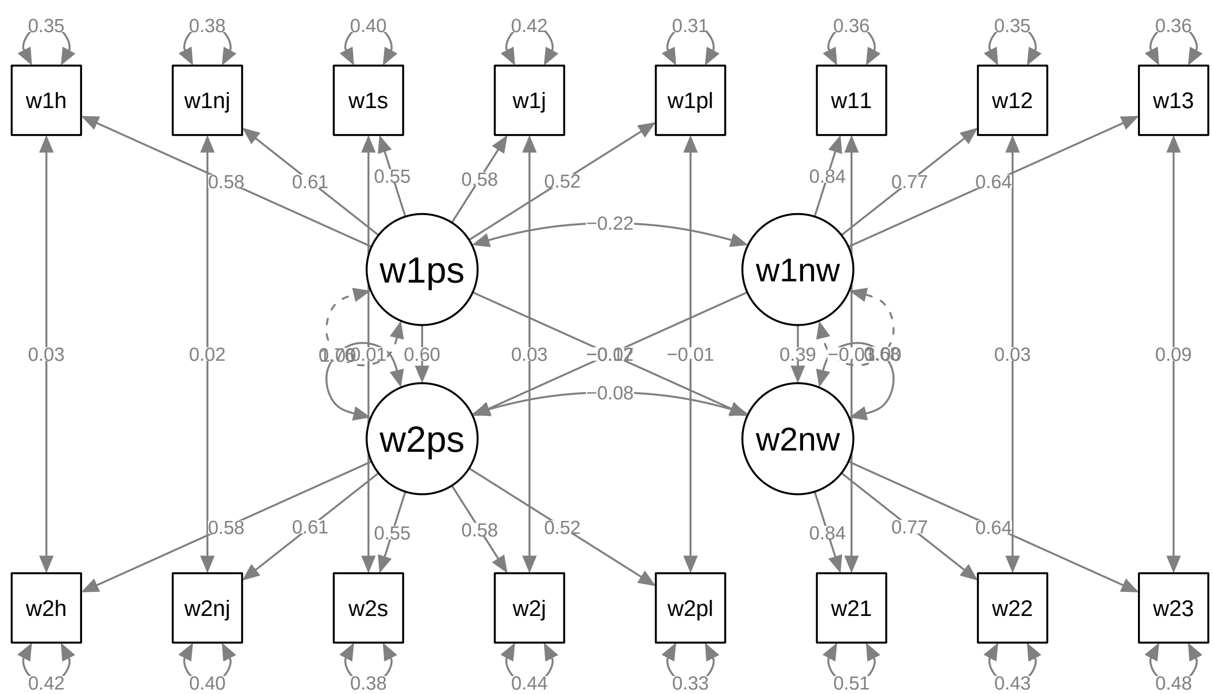
a. Estimate the parameters of the model in *Figure 5* (i.e., including standardized parameters) and evaluate the model fit.

Note that we will:

- use common labels for loadings to impose longitudinal equality constraints
- not use marker identification (i.e., the first loading is marker by default), hence `NA` removes the marker constraint

```
# Model syntax.
model_ex_2_clp <- "
    # Measurement part for first latent variable at both time points.
    w1posaff =~ NA * w1happy + l1 * w1happy + l2 * w1enjoy + l3 * w1satis + l4 * w1joyful + l5 * w1please
    w2posaff =~ NA * w2happy + l1 * w2happy + l2 * w2enjoy + l3 * w2satis + l4 * w2joyful + l5 * w2please
```

```
    # Measurement part for second latent variable at both time points.
    w1unw =~ NA * w1unw1  + l6 * w1unw1  + l7 * w1unw2 +  l8 * w1unw3
    w2unw =~ NA * w2unw1  + l6 * w2unw1  + l7 * w2unw2 +  l8 * w2unw3

    # Constrain exogenous factor variance at first occasion for identification.
    w1posaff ~~ 1 * w1posaff
    w1unw ~~ 1 * w1unw

    # Cross-lagged and auto-regressive effects.
    w2posaff ~ w1posaff + w1unw
    w2unw ~ w1unw + w1posaff

    # Correlated measurement residuals for first latent variable.
    w1happy ~~ w2happy
    w1enjoy ~~ w2enjoy
    w1satis ~~ w2satis
    w1joyful ~~ w2joyful
    w1please ~~ w2please

    # Correlated measurement residuals for second latent variable.
    w1unw1 ~~ w2unw1
    w1unw2 ~~ w2unw2
    w1unw3 ~~ w2unw3
"

# Fit model.
model_ex_2_clp_fit <- sem(model_ex_2_clp, data = data_ex_2)

# Visualize the model.
semPaths(model_ex_2_clp_fit, what = "paths", whatLabels = "est")
```



```r
# Model summary.
summary(model_ex_2_clp_fit, standardized = TRUE)
```

```
## lavaan 0.6-12 ended normally after 36 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of model parameters                        48
##   Number of equality constraints                     8
##
##   Number of observations                           574
##
## Model Test User Model:
##
##   Test statistic                               226.516
##   Degrees of freedom                                96
##   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
##   w1posaff =~
##     w1happy   (l1)    0.584    0.028   20.745    0.000    0.584    0.705
##     w1enjoy   (l2)    0.607    0.029   21.020    0.000    0.607    0.702
##     w1satis   (l3)    0.548    0.027   19.974    0.000    0.548    0.656
##     w1joyful  (l4)    0.578    0.029   19.974    0.000    0.578    0.666
##     w1please  (l5)    0.516    0.025   20.641    0.000    0.516    0.680
##   w2posaff =~
##     w2happy   (l1)    0.584    0.028   20.745    0.000    0.621    0.691
##     w2enjoy   (l2)    0.607    0.029   21.020    0.000    0.645    0.715
##     w2satis   (l3)    0.548    0.027   19.974    0.000    0.583    0.686
##     w2joyful  (l4)    0.578    0.029   19.974    0.000    0.614    0.681
##     w2please  (l5)    0.516    0.025   20.641    0.000    0.549    0.691
##   w1unw =~
##     w1unw1    (l6)    0.837    0.036   22.979    0.000    0.837    0.814
##     w1unw2    (l7)    0.773    0.035   22.255    0.000    0.773    0.795
##     w1unw3    (l8)    0.635    0.031   20.333    0.000    0.635    0.728
##   w2unw =~
##     w2unw1    (l6)    0.837    0.036   22.979    0.000    0.789    0.742
##     w2unw2    (l7)    0.773    0.035   22.255    0.000    0.729    0.744
##     w2unw3    (l8)    0.635    0.031   20.333    0.000    0.599    0.652
##
## Regressions:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   w2posaff ~
##     w1posaff          0.600    0.050   11.899    0.000    0.565    0.565
```

```
##     w1unw            -0.020    0.049   -0.406    0.685   -0.019   -0.019
##   w2unw ~
##     w1unw             0.390    0.048    8.169    0.000    0.414    0.414
##     w1posaff         -0.172    0.048   -3.598    0.000   -0.182   -0.182
##
## Covariances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##  .w1happy ~~
##    .w2happy           0.034    0.019    1.780    0.075    0.034    0.090
##  .w1enjoy ~~
##    .w2enjoy           0.016    0.020    0.789    0.430    0.016    0.040
##  .w1satis ~~
##    .w2satis           0.009    0.019    0.484    0.628    0.009    0.024
##  .w1joyful ~~
##    .w2joyful          0.028    0.021    1.323    0.186    0.028    0.065
##  .w1please ~~
##    .w2please         -0.010    0.016   -0.650    0.516   -0.010   -0.032
##  .w1unw1 ~~
##    .w2unw1           -0.034    0.027   -1.262    0.207   -0.034   -0.080
##  .w1unw2 ~~
##    .w2unw2            0.034    0.024    1.426    0.154    0.034    0.087
##  .w1unw3 ~~
##    .w2unw3            0.095    0.022    4.343    0.000    0.095    0.227
##   w1posaff ~~
##     w1unw            -0.220    0.049   -4.513    0.000   -0.220   -0.220
##  .w2posaff ~~
##    .w2unw            -0.082    0.043   -1.889    0.059   -0.114   -0.114
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##     w1posaff          1.000                               1.000    1.000
##     w1unw             1.000                               1.000    1.000
##    .w1happy           0.346    0.025   13.624    0.000    0.346    0.503
##    .w1enjoy           0.378    0.028   13.683    0.000    0.378    0.507
##    .w1satis           0.397    0.028   14.428    0.000    0.397    0.569
##    .w1joyful          0.418    0.029   14.283    0.000    0.418    0.556
##    .w1please          0.310    0.022   14.068    0.000    0.310    0.538
##    .w2happy           0.421    0.030   13.995    0.000    0.421    0.522
##    .w2enjoy           0.399    0.029   13.562    0.000    0.399    0.489
##    .w2satis           0.382    0.027   14.063    0.000    0.382    0.529
##    .w2joyful          0.436    0.031   14.153    0.000    0.436    0.536
##    .w2please          0.330    0.024   13.985    0.000    0.330    0.522
##    .w1unw1            0.356    0.036    9.803    0.000    0.356    0.337
##    .w1unw2            0.348    0.032   10.717    0.000    0.348    0.368
##    .w1unw3            0.358    0.027   13.067    0.000    0.358    0.470
##    .w2unw1            0.507    0.046   11.093    0.000    0.507    0.449
##    .w2unw2            0.428    0.038   11.123    0.000    0.428    0.446
##    .w2unw3            0.484    0.036   13.632    0.000    0.484    0.574
##    .w2posaff          0.763    0.084    9.044    0.000    0.676    0.676
##    .w2unw             0.678    0.076    8.873    0.000    0.762    0.762
```

```r
# Fit measures.
fitMeasures(model_ex_2_clp_fit, fit.measures = fit_indices)
```

```
##       chisq          df      pvalue         cfi         tli       rmsea rmsea.pvalue
##     226.516      96.000       0.000       0.959       0.948       0.049       0.593
##        srmr
##       0.042
```

The model has reasonably good fit to the data.

b. What is the size of the auto-regressive and cross-lagged standardized effects in this model?

We can look at the summary output for our model fit, under the `Regressions` heading.

```
# Regressions:
#                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
#   w2posaff ~
#     w1posaff          0.600    0.050   11.899    0.000    0.565    0.565
#     w1unw            -0.020    0.049   -0.406    0.685   -0.019   -0.019
#   w2unw ~
#     w1unw             0.390    0.048    8.169    0.000    0.414    0.414
#     w1posaff         -0.172    0.048   -3.598    0.000   -0.182   -0.182
```

## Exercise 3

Using the dataset `health.dat`:

a. Estimate for the six repeated measurements the trait-state-error model, as proposed by Kenny & Zautra (1995). You can use as starting point the model depicted in *Figure 6*, but note that it should be extended to include `y1` to `y6`.

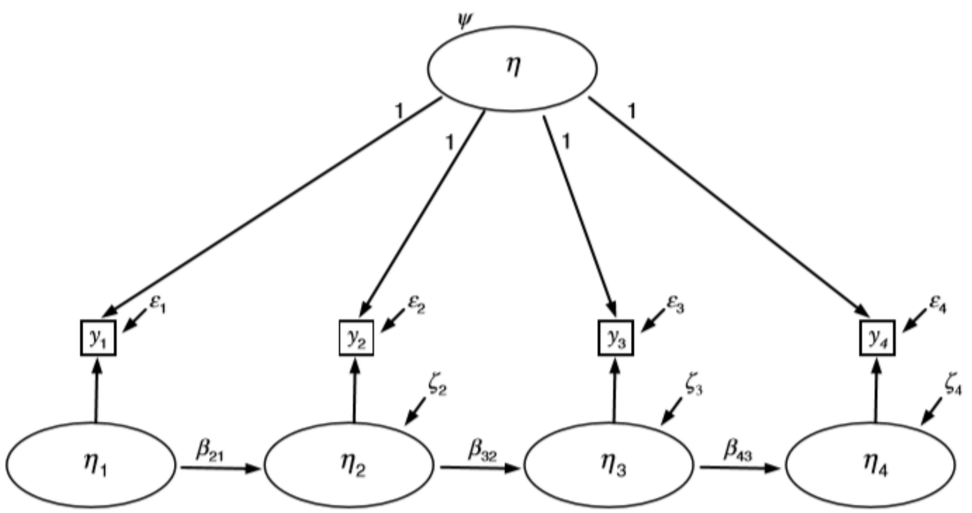



Figure 6: Example of repeated measures model.

Note. The latent *trait-state-error* model is equivalent to the univariate *STARTS* model discussed during the lecture. In a nutshell, the *STARTS* model consists of:

- stable trait (i.e., the "trait" in the *trait-state-error* terminology)
- auto-regressive trait (i.e., the "state" in the *trait-state-error* terminology)
- state (i.e., the "error" in the *trait-state-error* terminology), which, in the univariate *STARTS* model, also contains error as mentioned in the lecture

Load the data.

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

# Load data.
data_ex_3 <- read.table("health.dat")

# Inspect the data.
View(data_ex_3)
```

Set the variable names (i.e., same names as we used for *Exercise 1*).

```
# Set the names.
names(data_ex_3) <- variable_ex_1_names

# List variables.
```

```r
str(data_ex_3)
```

```
## 'data.frame':    5335 obs. of  37 variables:
##  $ age    : num  55.1 63.3 58.6 62.3 59.7 ...
##  $ srh1   : num  3.22 2.94 2.06 3.1 5.04 ...
##  $ srh2   : num  2.845 0.475 2.981 5.187 4.843 ...
##  $ srh3   : num  1.581 0.666 3.397 3.966 3.423 ...
##  $ srh4   : num  1.25 3.08 2.93 3.11 4.04 ...
##  $ srh5   : num  2.11 3.31 2.91 3.39 3.68 ...
##  $ srh6   : num  0.717 3.427 2.211 4.089 2.891 ...
##  $ bmi1   : num  27.1 24.7 13.9 23.7 23.8 ...
##  $ bmi2   : num  29.4 27.1 12.5 24.7 25.6 ...
##  $ bmi3   : num  29.7 27.9 13.4 24.2 25.4 ...
##  $ bmi4   : num  29.2 25.3 15.8 26.1 25.3 ...
##  $ bmi5   : num  27.9 26.7 15.9 25.4 28.2 ...
##  $ bmi6   : num  27.8 28 15.4 24.8 31.2 ...
##  $ cesdna1: num  0.743 0.1649 1.8722 1.3137 0.0949 ...
##  $ cesdpa1: num  0.4369 -0.0335 1.4751 0.4012 -0.8426 ...
##  $ cesdso1: num  1.01 1.55 2.19 1.51 1.03 ...
##  $ cesdna2: num  0.128 1.341 1.135 0.235 -0.114 ...
##  $ cesdpa2: num  0.0388 0.3245 0.5635 0.8914 -0.2595 ...
##  $ cesdso2: num  0.566 1.446 0.898 1.029 0.121 ...
##  $ cesdna3: num  1.282 0.646 1.654 1.12 -0.337 ...
##  $ cesdpa3: num  0.0045 0.1149 0.9773 0.0697 -0.5989 ...
##  $ cesdso3: num  0.982 1.147 0.544 1.091 -0.223 ...
##  $ cesdna4: num  0.766 0.452 0.947 0.9 -0.21 ...
##  $ cesdpa4: num  1.068 0.43 1.138 -0.281 -0.22 ...
##  $ cesdso4: num  1.345 1.921 0.78 1.28 0.534 ...
##  $ cesdna5: num  1.663 0.486 1.04 1.849 -0.978 ...
##  $ cesdpa5: num  0.4749 0.0759 0.854 0.3355 -0.3516 ...
##  $ cesdso5: num  1.149 1.395 1.149 0.909 0.309 ...
##  $ cesdna6: num  0.775 -0.657 1.105 1.009 0.291 ...
##  $ cesdpa6: num  0.929 -0.581 0.821 0.586 0.857 ...
##  $ cesdso6: num  0.522 1.077 1.341 1.256 0.957 ...
##  $ diab1  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ diab2  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ diab3  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ diab4  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ diab5  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ diab6  : int  0 0 0 0 0 0 0 0 0 0 ...
```

To construct a single observed measurement per wave, calculate the average score of the three indicators per wave, as follows:

```r
# Add the average scores per wave as follows.
data_ex_3$cesd1 = with(data_ex_3, (cesdna1 + cesdpa1 + cesdso1) / 3)
data_ex_3$cesd2 = with(data_ex_3, (cesdna2 + cesdpa2 + cesdso2) / 3)
data_ex_3$cesd3 = with(data_ex_3, (cesdna3 + cesdpa3 + cesdso3) / 3)
data_ex_3$cesd4 = with(data_ex_3, (cesdna4 + cesdpa4 + cesdso4) / 3)
data_ex_3$cesd5 = with(data_ex_3, (cesdna5 + cesdpa5 + cesdso5) / 3)
data_ex_3$cesd6 = with(data_ex_3, (cesdna6 + cesdpa6 + cesdso6) / 3)
```

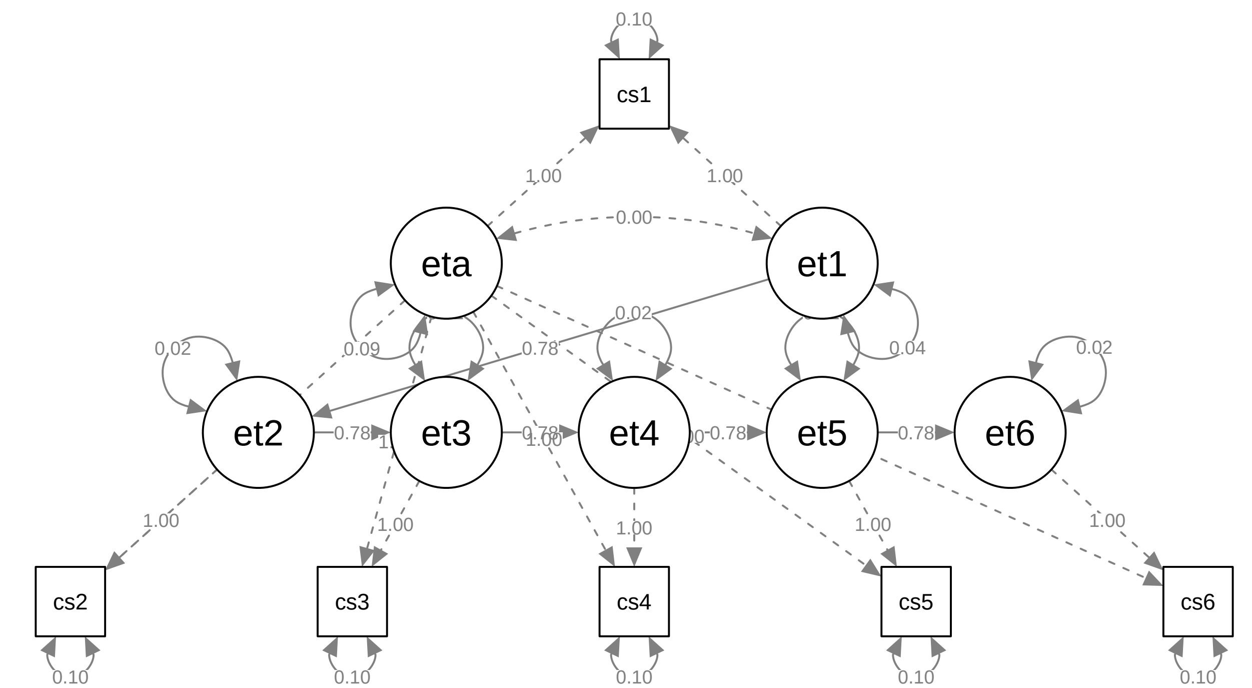
The model follows that shown in *Figure 2* and, specifies a single trait factor with the loading for each indicator set equal to 1 and a latent state factor with a single loading set equal to 1 at each occasion. Each state factor is regressed on the state factor from the prior time point. Several longitudinal equality constraints should be imposed (i.e., stationarity):

- all auto-regressive coefficients are set equal
- all state factor residuals except for the first state factor are set equal
- all measurement residuals are set equal

The trait variance and the state factor variance at $t_1$ are free to be estimated, and $\eta$ and $\eta_1$ are assumed independent.

```
# Model syntax.
model_ex_3_lts <- "
    # Trait factor.
    eta =~ 1 * cesd1 + 1 * cesd2 + 1 * cesd3 + 1 * cesd4 + 1 * cesd5 + 1 * cesd6

    # State factors.
    eta1 =~ cesd1
    eta2 =~ cesd2
    eta3 =~ cesd3
    eta4 =~ cesd4
    eta5 =~ cesd5
    eta6 =~ cesd6

    # Auto-regressive paths.
    eta2 ~ b * eta1
    eta3 ~ b * eta2
    eta4 ~ b * eta3
    eta5 ~ b * eta4
    eta6 ~ b * eta5

    # State factor residuals.
    # The variance of exogenous variable is model parameter.
    eta1 ~~ p2 * eta1
    eta2 ~~ p3 * eta2
    eta3 ~~ p3 * eta3
    eta4 ~~ p3 * eta4
    eta5 ~~ p3 * eta5
    eta6 ~~ p3 * eta6

    # Measurement residuals.
    cesd1 ~~ a * cesd1
    cesd2 ~~ a * cesd2
    cesd3 ~~ a * cesd3
    cesd4 ~~ a * cesd4
    cesd5 ~~ a * cesd5
    cesd6 ~~ a * cesd6

    # Exogenous covariances are set to zero.
    eta ~~ 0 * eta1
```

```r
"

# Fit model.
model_ex_3_lts_fit <- sem(model_ex_3_lts, data = data_ex_3)

# Visualize the model.
semPaths(model_ex_3_lts_fit, what = "paths", whatLabels = "est")
```




```r
# Model summary.
summary(model_ex_3_lts_fit, standardized = TRUE)
```

```
## lavaan 0.6-12 ended normally after 34 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of model parameters                        18
##   Number of equality constraints                    13
##
##   Number of observations                          5335
##
## Model Test User Model:
##
##   Test statistic                                81.074
##   Degrees of freedom                                16
##   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
##   eta =~
##     cesd1             1.000                               0.302    0.634
##     cesd2             1.000                               0.302    0.630
##     cesd3             1.000                               0.302    0.627
##     cesd4             1.000                               0.302    0.626
##     cesd5             1.000                               0.302    0.625
##     cesd6             1.000                               0.302    0.624
##   eta1 =~
##     cesd1             1.000                               0.195    0.410
##   eta2 =~
##     cesd2             1.000                               0.203    0.423
##   eta3 =~
##     cesd3             1.000                               0.208    0.431
##   eta4 =~
##     cesd4             1.000                               0.211    0.436
##   eta5 =~
##     cesd5             1.000                               0.212    0.439
##   eta6 =~
##     cesd6             1.000                               0.213    0.440
##
## Regressions:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   eta2 ~
##     eta1       (b)    0.778    0.077   10.147    0.000    0.748    0.748
##   eta3 ~
##     eta2       (b)    0.778    0.077   10.147    0.000    0.760    0.760
##   eta4 ~
##     eta3       (b)    0.778    0.077   10.147    0.000    0.768    0.768
##   eta5 ~
##     eta4       (b)    0.778    0.077   10.147    0.000    0.772    0.772
##   eta6 ~
##     eta5       (b)    0.778    0.077   10.147    0.000    0.774    0.774
##
## Covariances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   eta ~~
##     eta1              0.000                               0.000    0.000
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##     eta1      (p2)    0.038    0.008    5.030    0.000    1.000    1.000
##    .eta2      (p3)    0.018    0.003    6.367    0.000    0.441    0.441
##    .eta3      (p3)    0.018    0.003    6.367    0.000    0.422    0.422
##    .eta4      (p3)    0.018    0.003    6.367    0.000    0.411    0.411
##    .eta5      (p3)    0.018    0.003    6.367    0.000    0.404    0.404
##    .eta6      (p3)    0.018    0.003    6.367    0.000    0.401    0.401
##    .cesd1      (a)    0.098    0.002   39.735    0.000    0.098    0.430
##    .cesd2      (a)    0.098    0.002   39.735    0.000    0.098    0.424
```

```
##    .cesd3      (a)    0.098    0.002   39.735    0.000    0.098    0.421
##    .cesd4      (a)    0.098    0.002   39.735    0.000    0.098    0.419
##    .cesd5      (a)    0.098    0.002   39.735    0.000    0.098    0.418
##    .cesd6      (a)    0.098    0.002   39.735    0.000    0.098    0.417
##     eta               0.091    0.009   10.441    0.000    1.000    1.000
```

```r
# Fit measures.
fitMeasures(model_ex_3_lts_fit, fit.measures = fit_indices)
```

```
##        chisq           df       pvalue          cfi          tli        rmsea rmsea.pvalue
##       81.074       16.000        0.000        0.995        0.995        0.028        1.000
##         srmr
##        0.030
```

b. What is the fit of this model?

The model seems to fit the data reasonably well.

c. Verify the calculations reported by Newsom for this model:
   - proportion of trait variance equal to .43
   - proportion of state variance equal to .10
   - proportion of error variance equal to .47

We extract and calculate the proportions of variance as follows:

```r
# First take a look at the coefficients.
coefficients <- coef(model_ex_3_lts_fit)

# Let's store the names so we can quickly search by name.
names <- names(coefficients)

# Store all variances
variance_trait <- as.numeric(coefficients["eta~~eta"])
variance_state <- mean(coefficients[grepl("^p[0-9]$", names)])
variance_error <- mean(coefficients[grepl("^a$", names)])

# Print the variances.
c(
    trait = variance_trait,
    state = variance_state,
    error = variance_error
)
```

```
##      trait      state      error
## 0.09133696 0.02152825 0.09774900
```

```r
# Proportion of trait variance.
variance_trait / sum(variance_trait, variance_state, variance_error)
```

```
## [1] 0.4336695
```

```r
# Proportion of state variance.
variance_state / sum(variance_trait, variance_state, variance_error)
```

```
## [1] 0.1022165
```

```r
# Proportion of error variance.
variance_error / sum(variance_trait, variance_state, variance_error)
```

```
## [1] 0.464114
```

This, indeed, matches what Newsom reported.