Chapter 3: P-technique

Stephanie T. Lane & Kathleen M. Gates

Setting up the environment.

Make sure you have the following packages installed:

- mvtnorm
- nFactors
- lavaan
- ggplot2

Generating Data

Here we generate the simulated data used in Chapter 3.

Defining the matrices and parameters.

```
time      <- 125                   # length of time, T
psi       <- matrix(c(2.77, 2.47,   # factor covariance matrix
                      2.47, 8.40),
                    ncol = 2,
                    byrow = T)
print(psi)
```

```
##      [,1] [,2]
## [1,] 2.77 2.47
## [2,] 2.47 8.40
```

```
lambda    <- matrix(c(1, 0,         # loading matrix
                      2, 0,
                      1, 0,
                      0, 1,
                      0, 2,
                      0, 1),
                      ncol = 2,
                      byrow = TRUE)
print(lambda)
```

```
##      [,1] [,2]
## [1,]    1    0
## [2,]    2    0
## [3,]    1    0
## [4,]    0    1
## [5,]    0    2
## [6,]    0    1
```

```
theta     <- diag(.5,              # measurement error variance
                  ncol = 6,
```

```r
                  nrow = 6)
print(theta)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]  0.5  0.0  0.0  0.0  0.0  0.0
## [2,]  0.0  0.5  0.0  0.0  0.0  0.0
## [3,]  0.0  0.0  0.5  0.0  0.0  0.0
## [4,]  0.0  0.0  0.0  0.5  0.0  0.0
## [5,]  0.0  0.0  0.0  0.0  0.5  0.0
## [6,]  0.0  0.0  0.0  0.0  0.0  0.5
```

Generating the observed series.

We'll use the matrices defined above to generate the data.

```r
set.seed(12345) # This way each time we run the simulation we'll get the same values.

#### Generate the measurement error ####
epsilon   <- mvtnorm::rmvnorm(time, # generate meas. error
                     mean = c(0, 0, 0, 0, 0, 0),
                     sigma = theta)
head(epsilon)
```

```
##            [,1]       [,2]        [,3]       [,4]        [,5]       [,6]
## [1,]  0.4140314  0.5016682 -0.07728912 -0.3206709  0.42842713 -1.2854890
## [2,]  0.4455470 -0.1952917 -0.20093128 -0.6500588 -0.08219961  1.2850337
## [3,]  0.2620735  0.3678486 -0.53070626  0.5776354 -0.62674941 -0.2344608
## [4,]  0.7924635  0.2112296  0.55127595  1.0293955 -0.45560900 -1.0982340
## [5,] -1.1297512  1.2763967 -0.34057612  0.4386748  0.43283667 -0.1147712
## [6,]  0.5740810  1.5533959  1.44899638  1.1543134  0.17979688  0.3473226
```

```r
#### Generate the latent factor series ####
zeta      <- mvtnorm::rmvnorm(time, # generate factor scores
                     mean = c(0, 0),
                     sigma = psi)
head(zeta)
```

```
##           [,1]       [,2]
## [1,]  0.778160  0.5350046
## [2,]  0.640249 -1.7507008
## [3,]  1.217280 -1.4066082
## [4,] -2.053411 -0.9779769
## [5,]  2.374971 -1.6090356
## [6,] -2.354461 -4.3294729
```

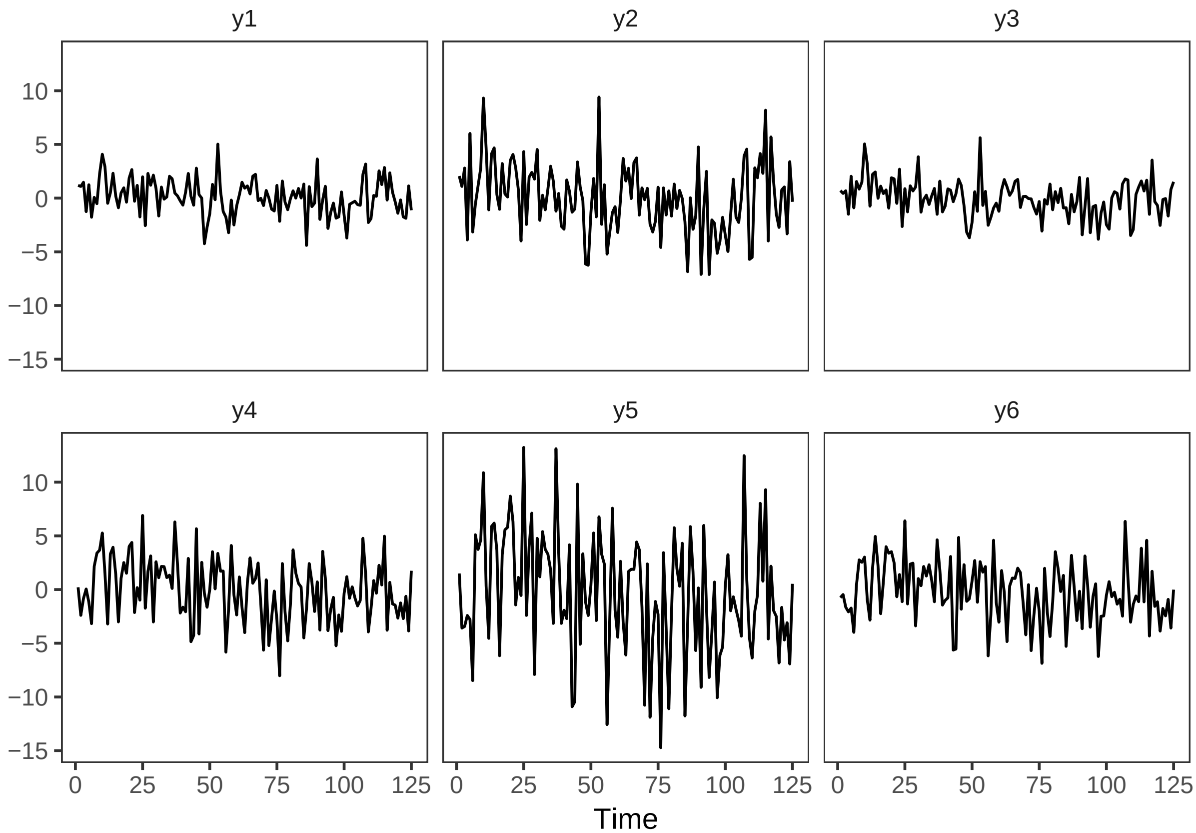
```r
#### Generate the observed variable series ####
y   <- matrix(0, nrow = time, ncol = 6)
for (p in 1:nrow(y)){
  y[p, ] <- lambda %*% zeta[p, ] + epsilon[p, ]
}
colnames(y) <- c("V1", "V2", "V3", "V4", "V5", "V6")
```

Plotting the observed time series

Let's take a look at the observed time series that we generated.

```r
library(ggplot2)
y2         <- as.data.frame(y)
y2$Time    <- seq(1:time)                     # Add a time vector
names(y2) <- c(paste0("y", seq(1:6)), "Time") # name variables
y2         <- reshape2::melt(y2, id = "Time")  # reshape

ggplot(y2, aes(Time, value)) + geom_line() +
  theme_bw() + ylab("") +
  facet_wrap(~ variable, ncol = 3) +
  theme(strip.background = element_blank(),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank())
```



Exploratory factor analysis: Manual

One Factor

For the one-factor solution, all loadings will be freely estimated. The variance of factor set to scalar 1. This scales the latent variable.

Alternatively we could scale according to the first observed variable by setting the lambda to equal 1. We'll show how to do this a bit later.

```
model <- '
FAC1 =~ NA*V1 + V2 + V3 + V4 + V5 + V6
FAC1 ~~ 1*FAC1
'
fit <- lavaan::cfa(model, as.data.frame(y))
lavaan::parameterEstimates(fit)
```

```
    lhs op  rhs    est    se      z pvalue ci.lower ci.upper
1  FAC1 =~   V1  1.479 0.118 12.515  0.000    1.248    1.711
2  FAC1 =~   V2  3.153 0.216 14.595  0.000    2.730    3.576
3  FAC1 =~   V3  1.550 0.118 13.155  0.000    1.319    1.781
4  FAC1 =~   V4  1.679 0.236  7.119  0.000    1.217    2.142
5  FAC1 =~   V5  3.261 0.466  6.994  0.000    2.347    4.175
6  FAC1 =~   V6  1.551 0.230  6.747  0.000    1.101    2.002
7  FAC1 ~~ FAC1  1.000 0.000     NA     NA    1.000    1.000
8    V1 ~~   V1  0.612 0.094  6.510  0.000    0.428    0.796
9    V2 ~~   V2  0.708 0.239  2.961  0.003    0.240    1.177
10   V3 ~~   V3  0.493 0.084  5.871  0.000    0.328    0.657
11   V4 ~~   V4  5.310 0.687  7.727  0.000    3.963    6.657
12   V5 ~~   V5 20.937 2.707  7.734  0.000   15.631   26.242
13   V6 ~~   V6  5.179 0.668  7.749  0.000    3.869    6.489
```

```
round(lavaan::fitMeasures(fit)
      [c("chisq", "df", "pvalue", "cfi", "tli", "rmsea", "srmr", "bic")], digits = 3)
```

```
   chisq       df   pvalue      cfi      tli    rmsea     srmr      bic
 555.322    9.000    0.000    0.475    0.125    0.697    0.237 3190.107
```

Two Factors, Orthogonal

One loading is fixed to zero in the second factor. We set the factor covariance matrix set to be the identity matrix. This sets the scale of the latent variables to have variance of 1.

```
model <- '
FAC1 =~ NA*V1 + V2 + V3 + V4 + V5 + V6
FAC2 =~ 0*V1  + V2 + V3 + V4 + V5 + V6
FAC1 ~~ 0*FAC2
FAC1 ~~ 1*FAC1
FAC2 ~~ 1*FAC2
'
fitlavaan <- lavaan::sem(model, as.data.frame(y))
lavaan::parameterEstimates(fitlavaan)
```

```
    lhs op  rhs    est    se      z pvalue ci.lower ci.upper
1  FAC1 =~   V1  1.472 0.118 12.432  0.000    1.240    1.705
2  FAC1 =~   V2  3.181 0.215 14.808  0.000    2.760    3.602
3  FAC1 =~   V3  1.559 0.118 13.227  0.000    1.328    1.790
4  FAC1 =~   V4  1.614 0.259  6.236  0.000    1.107    2.121
5  FAC1 =~   V5  3.134 0.514  6.093  0.000    2.126    4.142
6  FAC1 =~   V6  1.484 0.253  5.864  0.000    0.988    1.980
7  FAC2 =~   V1  0.000 0.000     NA     NA    0.000    0.000
8  FAC2 =~   V2 -0.075 0.174 -0.432  0.666   -0.417    0.267
9  FAC2 =~   V3 -0.142 0.099 -1.440  0.150   -0.335    0.051
10 FAC2 =~   V4  2.242 0.175 12.803  0.000    1.899    2.585
11 FAC2 =~   V5  4.580 0.339 13.517  0.000    3.916    5.244
12 FAC2 =~   V6  2.226 0.170 13.115  0.000    1.893    2.559
13 FAC1 ~~ FAC2  0.000 0.000     NA     NA    0.000    0.000
14 FAC1 ~~ FAC1  1.000 0.000     NA     NA    1.000    1.000
15 FAC2 ~~ FAC2  1.000 0.000     NA     NA    1.000    1.000
16   V1 ~~   V1  0.632 0.095  6.654  0.000    0.446    0.818
17   V2 ~~   V2  0.523 0.255  2.046  0.041    0.022    1.023
18   V3 ~~   V3  0.445 0.086  5.186  0.000    0.277    0.613
19   V4 ~~   V4  0.500 0.085  5.917  0.000    0.335    0.666
20   V5 ~~   V5  0.780 0.253  3.080  0.002    0.284    1.276
21   V6 ~~   V6  0.427 0.077  5.538  0.000    0.276    0.578
```

```
round(lavaan::fitMeasures(fitlavaan)
      [c("chisq", "df", "pvalue", "cfi", "tli", "rmsea", "srmr", "bic")], digits = 3)
```

```
   chisq       df   pvalue      cfi      tli    rmsea     srmr      bic
   5.232    4.000    0.264    0.999    0.996    0.050    0.004 2664.158
```

Two Factors, Oblique

Now, let's investigate the structure of these factors, or which observed variables load onto which latent variables.

```
model <- '
FAC1 =~ NA*V1 + V2 + V3 + V4 + V5 + V6
FAC2 =~ V1 + V2 + 0*V3 + V4 + V5 + V6
FAC1 ~~ 1*FAC1
FAC2 ~~ 1*FAC2 + FAC1
'
fit <- lavaan::cfa(model, as.data.frame(y), std.lv=F, std.ov=F)
lavaan::parameterEstimates(fit)
```

```
    lhs op  rhs     est     se      z pvalue ci.lower ci.upper
1  FAC1 =~   V1   0.475  0.119  3.990  0.000    0.242    0.709
2  FAC1 =~   V2   1.591  1.093  1.455  0.146   -0.552    3.733
3  FAC1 =~   V3   1.565  0.117 13.329  0.000    1.335    1.795
4  FAC1 =~   V4 -16.257 12.119 -1.341  0.180  -40.010    7.497
5  FAC1 =~   V5 -33.263 24.698 -1.347  0.178  -81.670   15.144
6  FAC1 =~   V6 -16.182 11.998 -1.349  0.177  -39.698    7.334
7  FAC2 =~   V1   1.000  0.000     NA     NA    1.000    1.000
8  FAC2 =~   V2   1.599  1.059  1.510  0.131   -0.477    3.674
9  FAC2 =~   V3   0.000  0.000     NA     NA    0.000    0.000
10 FAC2 =~   V4  17.820 12.059  1.478  0.139   -5.816   41.456
11 FAC2 =~   V5  36.294 24.576  1.477  0.140  -11.875   84.462
12 FAC2 =~   V6  17.616 11.939  1.475  0.140   -5.785   41.016
13 FAC1 ~~ FAC1   1.000  0.000     NA     NA    1.000    1.000
14 FAC2 ~~ FAC2   1.000  0.000     NA     NA    1.000    1.000
15 FAC1 ~~ FAC2   0.991  0.012 79.525  0.000    0.967    1.015
16   V1 ~~   V1   0.632  0.095  6.654  0.000    0.446    0.818
17   V2 ~~   V2   0.523  0.255  2.046  0.041    0.022    1.023
18   V3 ~~   V3   0.445  0.086  5.186  0.000    0.277    0.613
19   V4 ~~   V4   0.500  0.085  5.917  0.000    0.335    0.666
20   V5 ~~   V5   0.780  0.253  3.080  0.002    0.284    1.276
21   V6 ~~   V6   0.427  0.077  5.538  0.000    0.276    0.578
```

```
round(lavaan::fitMeasures(fit)
      [c("chisq", "df", "pvalue", "cfi", "tli", "rmsea", "srmr")], digits = 3)
```

```
 chisq     df pvalue    cfi    tli  rmsea   srmr
 5.232  4.000  0.264  0.999  0.996  0.050  0.004
```

The loadings for V1 to V3 onto FAC1 are all above 1.5 and significant. These variables do not appear to load onto FAC2.

Similarly the loadings for V4 to V6 on FAC2 are high and significant. These variables do not appear to load onto FAC1.

Final model. (Confirmatory analysis)

Remove the variables that had small loadings from the factor equations.

Here, we scale according to the first variable in each factor equation. This way we can see the covariance and variance of the latent factors.

```
model <- '
FAC1 =~ 1*V1 + V2 + V3
FAC2 =~ 1*V4 + V5 + V6
FAC1 ~~ FAC2
FAC1 ~~ FAC1
FAC2 ~~ FAC2
'
fitfinal <- lavaan::cfa(model, as.data.frame(y))
lavaan::parameterEstimates(fitfinal)
```

```
##     lhs op  rhs   est    se      z pvalue ci.lower ci.upper
## 1  FAC1 =~   V1 1.000 0.000     NA     NA    1.000    1.000
## 2  FAC1 =~   V2 2.180 0.123 17.688  0.000    1.939    2.422
## 3  FAC1 =~   V3 1.057 0.068 15.485  0.000    0.924    1.191
## 4  FAC2 =~   V4 1.000 0.000     NA     NA    1.000    1.000
## 5  FAC2 =~   V5 2.008 0.056 36.035  0.000    1.899    2.117
## 6  FAC2 =~   V6 0.968 0.031 31.173  0.000    0.907    1.029
## 7  FAC1 ~~ FAC2 2.186 0.435  5.023  0.000    1.333    3.038
## 8  FAC1 ~~ FAC1 2.154 0.349  6.181  0.000    1.471    2.837
## 9  FAC2 ~~ FAC2 7.633 1.028  7.425  0.000    5.618    9.648
## 10   V1 ~~   V1 0.646 0.097  6.662  0.000    0.456    0.836
## 11   V2 ~~   V2 0.410 0.247  1.655  0.098   -0.075    0.894
## 12   V3 ~~   V3 0.486 0.084  5.772  0.000    0.321    0.651
## 13   V4 ~~   V4 0.497 0.085  5.846  0.000    0.330    0.664
## 14   V5 ~~   V5 0.794 0.251  3.168  0.002    0.303    1.285
## 15   V6 ~~   V6 0.431 0.077  5.639  0.000    0.281    0.581
```

```
round(lavaan::fitMeasures(fitfinal)
      [c("chisq", "df", "pvalue", "cfi", "tli", "rmsea", "srmr")], digits = 3)
```

```
##  chisq     df pvalue    cfi    tli  rmsea   srmr
##  9.037  8.000  0.339  0.999  0.998  0.032  0.019
```

Exploratory Factor Analysis: Automatic

Exploratory factor analysis: One Factor

```r
# fitting model
fit       <- psych::fa(y, nfactors = 1, rotate = "varimax", fm = "ml")
# extracting fit indices
X2.model <- fit$STATISTIC
X2.null  <- fit$null.chisq
df.model <- fit$dof
df.null  <- fit$null.dof
RMSEA    <- fit$RMSEA[[1]]
TLI      <- fit$TLI
# computing fit indices not provided
CFI      <- ((X2.null - df.null) - (X2.model - df.model)) / (X2.null - df.null)
SRMR     <- sqrt(mean((fit$residual)^2))
# printing factor loading matrix
print(fit$loadings)
```

```

Loadings:
   ML1
V1 0.511
V2 0.545
V3 0.465
V4 0.970
V5 0.986
V6 0.971

                 ML1
SS loadings    3.630
Proportion Var 0.605
```

```r
# printing acceleration factor information
print(nFactors::nScree(fit$e.values)$Analysis[ ,c(1,7,8)])
```

```
  Eigenvalues Acc.factor     AF
1  4.26268323         NA (< AF)
2  1.35196691 1.75765916
3  0.19890976 1.04886964
4  0.09472225 0.06638182
5  0.05691656 0.01569043
6  0.03480130         NA
```

```r
# printing all fit measures
fits        <- c(X2.model, df.model, fit$PVAL, CFI, TLI, RMSEA, SRMR)
names(fits) <- c("X2", "df", "pval", "CFI", "TLI", "RMSEA", "SRMR")
print(round(fits, digits = 3))
```

```
     X2      df    pval     CFI     TLI   RMSEA    SRMR
303.056   9.000   0.000   0.708   0.511   0.521   0.325
```

Exploratory factor analysis: Two Factors, Orthogonal

```r
fit      <- psych::fa(y, nfactors = 2, rotate = "varimax", fm = "ml")
# extracting fit indices
X2.model <- fit$STATISTIC
X2.null  <- fit$null.chisq
df.model <- fit$dof
df.null  <- fit$null.dof
RMSEA    <- fit$RMSEA[[1]]
TLI      <- fit$TLI
# computing fit indices not provided
CFI      <- ((X2.null - df.null) - (X2.model - df.model)) / (X2.null - df.null)
SRMR     <- sqrt(mean((fit$residual)^2))
# printing factor loading matrix
print(fit$loadings)
```

```
Loadings:
   ML1   ML2
V1 0.294 0.829
V2 0.304 0.926
V3 0.228 0.891
V4 0.930 0.271
V5 0.955 0.253
V6 0.942 0.238

                 ML1   ML2
SS loadings    2.895 2.534
Proportion Var 0.482 0.422
Cumulative Var 0.482 0.905
```

```r
# printing acceleration factor information
print(nFactors::nScree(fit$e.values)$Analysis[ ,c(1,7,8)])
```

```
  Eigenvalues Acc.factor     AF
1  4.26268323         NA (< AF)
2  1.35196691 1.75765916
3  0.19890976 1.04886964
4  0.09472225 0.06638182
5  0.05691656 0.01569043
6  0.03480130         NA
```

```r
# printing all fit measures
fits        <- c(X2.model, df.model, fit$PVAL, CFI, TLI, RMSEA, SRMR)
names(fits) <- c("X2", "df", "pval", "CFI", "TLI", "RMSEA", "SRMR")
print(round(fits, digits = 3))
```

```
   X2    df  pval   CFI   TLI RMSEA  SRMR
5.015 4.000 0.286 0.999 0.996 0.049 0.049
```

Exploratory factor analysis using nFactors

We can easily explore the eigenvalues and acceleration factor with nFactors. This provides a couple more ways for us to evaluate the number of latent factors in our data.

```
print(nFactors::nScree(eigen(cor(y))$values)$Analysis[ ,c(1,7,8)])
```

```
##   Eigenvalues Acc.factor     AF
## 1  4.26268323         NA (< AF)
## 2  1.35196691 1.75765916
## 3  0.19890976 1.04886964
## 4  0.09472225 0.06638182
## 5  0.05691656 0.01569043
## 6  0.03480130         NA
```

Taken together, the results point to a 2-factor solution.

Exploratory factor analysis: Two Factors, Oblique

```r
fit      <- psych::fa(y, nfactors = 2, rotate = "promax", fm = "ml")
```

```
Loading required namespace: GPArotation
```

```r
# extracting fit indices
X2.model <- fit$STATISTIC
X2.null  <- fit$null.chisq
df.model <- fit$dof
df.null  <- fit$null.dof
RMSEA    <- fit$RMSEA[[1]]
TLI      <- fit$TLI
# computing fit indices not provided
CFI      <- ((X2.null - df.null) - (X2.model - df.model)) / (X2.null - df.null)
SRMR     <- sqrt(mean((fit$residual)^2))
# printing factor loading matrix
print(fit$loadings)
```

```
Loadings:
   ML1    ML2
V1         0.853
V2         0.961
V3         0.944
V4  0.954
V5  0.988
V6  0.979

                 ML1   ML2
SS loadings    2.849 2.543
Proportion Var 0.475 0.424
Cumulative Var 0.475 0.899
```

```r
# printing acceleration factor information
print(nFactors::nScree(fit$e.values)$Analysis[ ,c(1,7,8)])
```

```
  Eigenvalues Acc.factor     AF
1  4.26268323         NA (< AF)
2  1.35196691 1.75765916
3  0.19890976 1.04886964
4  0.09472225 0.06638182
5  0.05691656 0.01569043
6  0.03480130         NA
```

```r
# printing all fit measures
fits         <- c(X2.model, df.model, fit$PVAL, CFI, TLI, RMSEA, SRMR)
names(fits) <- c("X2", "df", "pval", "CFI", "TLI", "RMSEA", "SRMR")
print(round(fits, digits = 3))
```

```
   X2    df  pval   CFI   TLI RMSEA  SRMR
5.015 4.000 0.286 0.999 0.996 0.049 0.049
```

Run confirmatory analysis on simple structure

```r
# Here's an efficient way to write the model. This is helpful if you have lots of variables.
# Note that we get the same model as the final one posted above.
factor1 <- paste("FA1 =~",
                 colnames(y)[1:3],
                 collapse = "\n")
factor2 <- paste("FA2 =~",
                 colnames(y)[4:6],
                 collapse = "\n")
model <- paste(factor1, factor2,
               "FA1~~FA1 +FA2",
               "FA2~~FA2",
               sep = "\n")
fit <- lavaan::cfa(model, as.data.frame(y))
lavaan::parameterEstimates(fit)
```

```
   lhs op rhs   est    se      z pvalue ci.lower ci.upper
1  FA1 =~  V1 1.000 0.000     NA     NA    1.000    1.000
2  FA1 =~  V2 2.180 0.123 17.688  0.000    1.939    2.422
3  FA1 =~  V3 1.057 0.068 15.485  0.000    0.924    1.191
4  FA2 =~  V4 1.000 0.000     NA     NA    1.000    1.000
5  FA2 =~  V5 2.008 0.056 36.035  0.000    1.899    2.117
6  FA2 =~  V6 0.968 0.031 31.173  0.000    0.907    1.029
7  FA1 ~~ FA1 2.154 0.349  6.181  0.000    1.471    2.837
8  FA1 ~~ FA2 2.186 0.435  5.023  0.000    1.333    3.038
9  FA2 ~~ FA2 7.633 1.028  7.425  0.000    5.618    9.648
10  V1 ~~  V1 0.646 0.097  6.662  0.000    0.456    0.836
11  V2 ~~  V2 0.410 0.247  1.655  0.098   -0.075    0.894
12  V3 ~~  V3 0.486 0.084  5.772  0.000    0.321    0.651
13  V4 ~~  V4 0.497 0.085  5.846  0.000    0.330    0.664
14  V5 ~~  V5 0.794 0.251  3.168  0.002    0.303    1.285
15  V6 ~~  V6 0.431 0.077  5.639  0.000    0.281    0.581
```