Replication Instruction

The R code was developed using R version 4.4.1 (2024-06-14) and RStudio version 2024.04.2. This instruction uses N=100 in Table A1 (which is the full result of Table 1) as an example; the procedure applies to all other tables as well. This code is very tedious and complicated, running it is time consuming. Pay close attention to the bold sections, as they require careful consideration and thorough review. Please be patient. If you have any problems, please email me at bangquan@ucla.edu.

Required R Packages

The R packages needed for this replication are **lavaan** (version 0.6.19.) for structural equation modeling and **boot** for bootstrapping procedures.

Install.packages ("lavaan")
Install.packages ("boot")

Simulation Setup

1. Set up the covariance matrix to derive the residual variances (the diagonal of ψ), which will be used for simulation.

```
identity<-diag(1, 24,24)
phi<-matrix(c( 1, 0.3, 0.4,
               0.3, 1, 0.5,
               0.4, 0.5, 1
), 3,3)

ld<-matrix(c(0.65, 0.65, 0.7, 0.7, 0.7, 0.6, 0.6, 0.5,   0.5, 0, 0, 0, 0, 0, 0,0,  0, 0, 0, 0, 0, 0, 0, 0,
             0, 0, 0, 0, 0, 0.5, 0, 0,  0.6, 0.6, 0.6, 0.7, 0.7, 0.5, 0.5, 0.55, 0, 0, 0, 0.75, 0, 0, 0, 0,
             0, 0, 0, 0, 0, 0, 0, 0,  0, 0, 0, 0, 0, 0, 0, 0.45,  0.5, 0.5, 0.5, 0.6, 0.6, 0.6, 0.7, 0.7
), nrow=24, ncol=3)

ld2<-t(ld)
psi<-diag(1,24)-diag(ld %*% phi %*% ld2)
diag(psi)
```

Note that you don't have to run covariance matrix, because the results are included in step (2) for simulation.

2. Based on this covariance matrix we start setup the population and analysis models, we call them "popmodel" and "mis_model".

```
popmodel<-'
f1 =~ 0.65*x1 + 0.65*x2 + 0.7*x3 + 0.7*x4 + 0.7*x5 + 0.7*x6 + 0.6*x7 + 0.5*x8 + 0.5*x9
f2 =~ 0.6*x9 + 0.6*x10 + 0.6*x11 + 0.7*x12 + 0.7*x13 + 0.5*x14 + 0.5*x15 + 0.65*x16 + 0.5*x6
+ 0.75*x20
f3 =~ 0.5*x17 + 0.5*x18 + 0.5*x19 + 0.6*x20 + 0.6*x21 + 0.6*x22 + 0.7*x23 + 0.7*x24 +
0.45*x16
f3 ~~ 0.4*f1 + 0.5*f2
f1 ~~ 0.3*f2
f1 ~ 0*1
f2 ~ 0*1
f3 ~ 0*1

x1 ~~ 0.5775*x1
x2 ~~ 0.5775*x2
x3 ~~ 0.51*x3
x4 ~~ 0.51*x4
x5 ~~ 0.51*x5
x6 ~~ 0.05*x6
x7 ~~ 0.64*x7
x8 ~~ 0.21*x8
x9 ~~ 0.21*x9
x10 ~~  0.64*x10
x11 ~~  0.51*x11
x12 ~~ 0.51*x12
x13 ~~ 0.75*x13
x14 ~~ 0.75*x14
x15 ~~ 0.5775*x15
x16 ~~ 0.7975*x16
x17 ~~ 0.75*x17
x18 ~~ 0.75*x18
x19 ~~ 0.25*x19
x20 ~~ 0.64*x20
x21 ~~ 0.64*x21
x22 ~~ 0.64*x22
x23 ~~ 0.51*x23
```

```
x24 ~~ 0.51*x24
'


mis_model<-'
f1 =~ x1 + x2 + x3 + x4 + x5 + x6  + x7 + x8
f2 =~ x9 + x10 + x11 + x12 + x13 + x14 + x15 + x16
f3 =~ x17 + x18 + x19 + x20 + x21 + x22 + x23 + x24
f3 ~~ f1
f1 ~~ f2
'
```

3. Based on the population model, we conduct Monte Carlo simulation using R package 'Lavaan', with which we obtain simulated data.

```
set.seed(105)
mydata<-simulateData(popmodel, model.type = "sem", sample.nobs=100L)
fit1 <- sem(model = mis_model, data=mydata, meanstructure=FALSE, likelihood = "wishart", estimator = "ML")
```

Univariate LM Test

4. We begin by performing a univariate LM test, sorting the results, and selecting the top 12 values for further analysis.

```
modindices(fit1, minimum.value = 0, sort = TRUE)[1:12,]
```

```
> modindices(fit1, minimum.value = 0, sort = TRUE)[1:12,]
    lhs op rhs     mi    epc sepc.lv sepc.all sepc.nox
76   f2 =~  x6 92.098  3.061   2.360    2.859    2.859
102  f3 =~ x16 24.355  1.248   0.779    0.612    0.612
82   f2 =~ x20 20.773  0.810   0.624    0.532    0.532
55   f1 =~  x9 17.748  1.615   0.808    0.900    0.900
172  x4 ~~  x8 12.349  0.151   0.151    0.356    0.356
72   f2 =~  x2 11.601 -1.116  -0.860   -0.852   -0.852
176  x4 ~~ x12 11.393 -0.234  -0.234   -0.350   -0.350
171  x4 ~~  x7 10.763  0.234   0.234    0.330    0.330
191  x5 ~~  x8 10.058  0.123   0.123    0.322    0.322
303 x12 ~~ x15  9.999  0.229   0.229    0.334    0.334
170  x4 ~~  x6  9.644 -0.116  -0.116   -2.400   -2.400
73   f2 =~  x3  8.573 -0.853  -0.658   -0.687   -0.687
```

We use these 12 parameters to calculate their LM test chi-square statistics and p-values. Next, we define 'newpar' and run the lavTestScore() function to the obtain the univariate LM test results.

```
newpar = '
f2=~x6
f3=~x16
f2=~x20
f1=~x9
x4~~x8
f2=~x2
x4~~x12
x4~~x7
x5~~x8
x12~~ x15
x4~~x6
f2=~x3
'
```

```
lavTestScore(fit1, add=newpar)
```

```
        lhs op rhs     X2 df p.value
1    f2=~x6 ==   0 92.098  1   0.000
2   f3=~x16 ==   0 24.355  1   0.000
3   f2=~x20 ==   0 20.773  1   0.000
4    f1=~x9 ==   0 17.747  1   0.000
5    x4~~x8 ==   0 12.349  1   0.000
6    f2=~x2 ==   0 11.601  1   0.001
7   x4~~x12 ==   0 11.393  1   0.001
8    x4~~x7 ==   0 10.763  1   0.001
9    x5~~x8 ==   0 10.058  1   0.002
10 x12~~x15 ==   0  9.999  1   0.002
11   x6~~x4 ==   0  9.644  1   0.002
12   f2=~x3 ==   0  8.573  1   0.003
```

The table above displays the parameters along with their LM chi-square values and *p*-values.

Bootstrap LM Test

5. To perform the bootstrap LM test, we manually input two parameters at a time and execute the bootstrap resampling procedure. During each iteration, the model is updated by adding two parameters to 'newpar' and adjusting the values in **lavTestScore** accordingly. This process continues until all 12 parameters have been systematically incorporated into the analysis.

```
newpar = '
f2=~x6
f3=~x16
'



set.seed(105)
Data<-mydata
# Initialize the matrix to store bootstrap results
bin.1000 <- matrix(NA, nrow = 500, ncol = 2)

# Loop through bootstrap iterations
for (i in 1:500) {

  # Perform one bootstrap iteration
  boot.res <- tryCatch({
    # Generate bootstrap sample
    boot.idx <- sample.int(nrow(Data), replace = TRUE)
    Data.boot <- Data[boot.idx, ]

    # Fit the SEM model
    fit2 <- sem(model = mis_model, data = Data.boot, int.ov.free = FALSE, std.lv = FALSE,
estimator = "ML")

    # Calculate scores
    score1 <- lavTestScore(fit2, add = newpar)$uni[1, 4]
    score2 <- lavTestScore(fit2, add = newpar)$uni[1, 6]

    # Return scores
    list(score1 = score1, score2 = score2)
  }, error = function(e) {
    # If SEM model did not converge, return NULL
    NULL
```

```
  })

  # Check if boot.res is not NULL (i.e., SEM model converged)
  if (!is.null(boot.res)) {
    # Update the matrix with scores
    bin.1000[i, 1] <- boot.res$score1
    bin.1000[i, 2] <- boot.res$score2
  }
}

# Calculate the means of the scores
t1 <- mean(bin.1000[, 1], na.rm = TRUE)
t2 <- mean(bin.1000[, 2], na.rm = TRUE)
t1
t2
```

Note that we need to manually change these values, [1,4] and [1,6] one by one.

```
score1 <- lavTestScore(fit2, add = newpar)$uni[1, 4]
score2 <- lavTestScore(fit2, add = newpar)$uni[1, 6]
```

After running this code, we calculate t1 and t2 (the LM chi-square and p-value). Then, we update t1 and t2 to [2, 4] and [2, 6], respectively. Next, we rerun the code with these updated parameters. Afterward, we add (f2~x20) and (f2~ x20) and (f1~x9) and (f1~x9) to 'newpar' and execute the code again. This process is repeated until all parameters have been processed.

```
newpar = '
f2=~x6
f3=~x16
f2=~x20
f1=~x9'
```

We repeat this procedure until we finish those 12 parameters. We keep those statistically significant (p-value<0.05) parameters for bootstrap Wald test.

Bootstrap Wald Test

6. **To perform a bootstrap Wald test, we incorporate the bootstrap LM test suggested statistical parameters into the original analysis model, which we call it "w_model". Note that this is very tricky to include the right parameters into the "w_model". I would suggest following Table A1 to see which bootstrap LM test suggested**

parameters to include. To distinguish these newly added parameters, we denote them as b1, b2, b3 and so on. Note that parameters that are not statistically significant will not be included in 'w_model'. If you attempt to run the model, R will return 'NaN'. For replication purposes, we have prepared all the necessary code.

```
w_model<-'

f1 =~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + b4*x9
f2 =~ x9 + x10 + x11 + x12 + x13 + x14 + x15 + x16 + b1*x6 + b3*x20 + b6*x2
f3 =~ x17 + x18 + x19 + x20 + x21 + x22 + x23 + x24 + b2*x16
f3 ~~ f1
f1 ~~ f2
x4~~b7*x12
x4~~b8*x7
x12~~b10*x15
x6~~b11*x4
'
```

After setting up the "w_model," we include the model and perform a backward stepwise procedure, constraining b1 = 0, and then run the model. This provides the Wald test and its corresponding p-value. Next, we add another constraint, such as b2==0, and repeat this process until all bootstrap LM test-suggested parameters have been evaluated.

```
set.seed(105)

con='
b1==0'

Data<-mydata
# Initialize the matrix to store bootstrap results
bin.1000 <- matrix(NA, nrow = 500, ncol = 2)

# Loop through bootstrap iterations
for (i in 1:500) {

  # Perform one bootstrap iteration
  boot.res <- tryCatch({
    # Generate bootstrap sample
    boot.idx <- sample.int(nrow(Data), replace = TRUE)
    Data.boot <- Data[boot.idx, ]

    # Fit the SEM model
```

```
    fit2 <- sem(model = w_model, data = Data.boot, meanstructure = FALSE, likelihood =
"wishart", estimator = "ML")

    # Calculate Wald test statistics and p-values
    wald_stat <- lavTestWald(fit2, constraints = con)$stat[1]
    wald_p_value <- lavTestWald(fit2, constraints = con)$p.value[1]

    # Return statistics
    list(stat = wald_stat, p_value = wald_p_value)
  }, error = function(e) {
    # If SEM model did not converge, return NULL
    NULL
  })

  # Check if boot.res is not NULL (i.e., SEM model converged)
  if (!is.null(boot.res)) {
    # Update the matrix with Wald test statistics and p-values
    bin.1000[i, 1] <- boot.res$stat
    bin.1000[i, 2] <- boot.res$p_value
  }
}

# Calculate the means of the Wald test statistics and p-values
w1 <- mean(bin.1000[, 1], na.rm = TRUE)
w2 <- mean(bin.1000[, 2], na.rm = TRUE)
w1
w2
```

Likelihood Ratio Test

7. To perform the likelihood ratio test, we first set up a base model, referred to as "m_0," which is the original analysis model. Next, we create a series of nested models, each including an additional parameter suggested by the univariate LM test—specifically, the parameters listed in "newpar."

```
m_0<-'
f1 =~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8
f2 =~ x9 + x10 + x11 + x12 + x13 + x14 + x15 + x16
f3 =~ x17 + x18 + x19 + x20 + x21 + x22 + x23 + x24
f3 ~~ f1
f1 ~~ f2
'
```

```
lrt_0<-sem(model = m_0, data=mydata, meanstructure=FALSE, likelihood = "wishart", estimator
= "ML")

m_1<-'
f1 =~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8
f2 =~ x9 + x10 + x11 + x12 + x13 + x14 + x15 + x16 + x6
f3 =~ x17 + x18 + x19 + x20 + x21 + x22 + x23 + x24
f3 ~~ f1
f1 ~~ f2
'
lrt_1<-sem(model = m_1, data=mydata, meanstructure=FALSE, likelihood = "wishart", estimator
= "ML")

m_2<-'
f1 =~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8
f2 =~ x9 + x10 + x11 + x12 + x13 + x14 + x15 + x16 + x6
f3 =~ x17 + x18 + x19 + x20 + x21 + x22 + x23 + x24 + x16
f3 ~~ f1
f1 ~~ f2
'
lrt_2<-sem(model = m_2, data=mydata, meanstructure=FALSE, likelihood = "wishart", estimator
= "ML")
lrt_2


m_3<-'
f1 =~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8
f2 =~ x9 + x10 + x11 + x12 + x13 + x14 + x15 + x16 + x6 + x20
f3 =~ x17 + x18 + x19 + x20 + x21 + x22 + x23 + x24 + x16
f3 ~~ f1
f1 ~~ f2
'
lrt_3<-sem(model = m_3, data=mydata, meanstructure=FALSE, likelihood = "wishart", estimator
= "ML")

m_4<-'
f1 =~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9
f2 =~ x9 + x10 + x11 + x12 + x13 + x14 + x15 + x16 + x6 + x20
f3 =~ x17 + x18 + x19 + x20 + x21 + x22 + x23 + x24 + x16
f3 ~~ f1
f1 ~~ f2
'
lrt_4<-sem(model = m_4, data=mydata, meanstructure=FALSE, likelihood = "wishart", estimator
= "ML")

m_5<-'
f1 =~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9
f2 =~ x9 + x10 + x11 + x12 + x13 + x14 + x15 + x16 + x6 + x20
```

```
f3 =~ x17 + x18 + x19 + x20 + x21 + x22 + x23 + x24 + x16
f3 ~~ f1
f1 ~~ f2
x4~~x8
'
lrt_5<-sem(model = m_5, data=mydata, meanstructure=FALSE, likelihood = "wishart", estimator
= "ML")

m_6<-'
f1 =~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9
f2 =~ x9 + x10 + x11 + x12 + x13 + x14 + x15 + x16 + x6 + x20 + x2
f3 =~ x17 + x18 + x19 + x20 + x21 + x22 + x23 + x24 + x16
f3 ~~ f1
f1 ~~ f2
x4~~x8
'
lrt_6<-sem(model = m_6, data=mydata, meanstructure=FALSE, likelihood = "wishart", estimator
= "ML")

m_7<-'
f1 =~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9
f2 =~ x9 + x10 + x11 + x12 + x13 + x14 + x15 + x16 + x6 + x20 + x2
f3 =~ x17 + x18 + x19 + x20 + x21 + x22 + x23 + x24 + x16
f3 ~~ f1
f1 ~~ f2
x4~~x8
x4~~x12

'
lrt_7<-sem(model = m_7, data=mydata, meanstructure=FALSE, likelihood = "wishart", estimator
= "ML")

m_8<-'
f1 =~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9
f2 =~ x9 + x10 + x11 + x12 + x13 + x14 + x15 + x16 + x6 + x20 + x2
f3 =~ x17 + x18 + x19 + x20 + x21 + x22 + x23 + x24 + x16
f3 ~~ f1
f1 ~~ f2
x4~~x8
x4~~x12
x4~~x7
'



lrt_8<-sem(model = m_8, data=mydata, meanstructure=FALSE, likelihood = "wishart", estimator
= "ML")

m_9<-'
```

```
f1 =~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9
f2 =~ x9 + x10 + x11 + x12 + x13 + x14 + x15 + x16 + x6 + x20 + x2
f3 =~ x17 + x18 + x19 + x20 + x21 + x22 + x23 + x24 + x16
f3 ~~ f1
f1 ~~ f2
x4~~x8
x4~~x12
x4~~x7
x5~~x8
'


lrt_9<-sem(model = m_9, data=mydata, meanstructure=FALSE, likelihood = "wishart", estimator
= "ML")

m_10<-'
f1 =~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9
f2 =~ x9 + x10 + x11 + x12 + x13 + x14 + x15 + x16 + x6 + x20 + x2
f3 =~ x17 + x18 + x19 + x20 + x21 + x22 + x23 + x24 + x16
f3 ~~ f1
f1 ~~ f2
x4~~x8
x4~~x12
x4~~x7
x5~~x8
x12~~x15

'

lrt_10<-sem(model = m_10, data=mydata, meanstructure=FALSE, likelihood = "wishart",
estimator = "ML")
lrt_10

m_11<-'
f1 =~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9
f2 =~ x9 + x10 + x11 + x12 + x13 + x14 + x15 + x16 + x6 + x20 + x2
f3 =~ x17 + x18 + x19 + x20 + x21 + x22 + x23 + x24 + x16
f3 ~~ f1
f1 ~~ f2

x4~~x8
x4~~x12
x4~~x7
x5~~x8
x12~~x15
x6~~x4

 '
```

```
lrt_11<-sem(model = m_11, data=mydata, meanstructure=FALSE, likelihood = "wishart",
estimator = "ML")
lrt_11


m_12<-'
f1 =~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9
f2 =~ x9 + x10 + x11 + x12 + x13 + x14 + x15 + x16 + x6 + x20 + x2 + x3
f3 =~ x17 + x18 + x19 + x20 + x21 + x22 + x23 + x24 + x16
f3 ~~ f1
f1 ~~ f2

x4~~x8
x4~~x12
x4~~x7
x5~~x8
x12~~x15
x6~~x4

'

lrt_12<-sem(model = m_12, data=mydata, meanstructure=FALSE, likelihood = "wishart",
estimator = "ML")
lrt_12

lavTestLRT(lrt_0, lrt_1, lrt_2, lrt_3, lrt_4, lrt_5, lrt_6, lrt_7, lrt_8, lrt_9, lrt_10, lrt_11, lrt_12)
```

```
Chi-Squared Difference Test

        Df    AIC    BIC  Chisq Chisq diff   RMSEA Df diff Pr(>Chisq)
lrt_12 237 5719.3 5883.4 254.62
lrt_11 238 5717.5 5879.0 254.87      0.246 0.00000       1    0.62023
lrt_10 239 5720.5 5879.4 259.81      4.944 0.19860       1    0.02618 *
lrt_9  240 5723.4 5879.7 264.64      4.835 0.19584       1    0.02788 *
lrt_8  241 5724.2 5878.0 267.48      2.839 0.13560       1    0.09202 .
lrt_7  242 5727.3 5878.4 272.45      4.967 0.19917       1    0.02584 *
lrt_6  243 5730.1 5878.6 277.23      4.781 0.19445       1    0.02877 *
lrt_5  244 5734.2 5880.1 283.32      6.093 0.22567       1    0.01357 *
lrt_4  245 5732.9 5876.2 283.98      0.660 0.00000       1    0.41640
lrt_3  246 5760.7 5901.3 313.42     29.440 0.53329       1  5.768e-08 ***
lrt_2  247 5782.6 5920.6 337.11     23.682 0.47626       1  1.136e-06 ***
lrt_1  248 5806.9 5942.4 363.19     26.080 0.50080       1  3.276e-07 ***
lrt_0  249 5871.8 6004.7 429.41     66.223 0.80761       1  4.026e-16 ***
```

Note that the results are reported in reverse order. We need to re-order them from lrt_0 to lrt_12.

lrt_0	249	5871.8	6004.7	429.41	66.223	1	4.03E-16 ***
lrt_1	248	5806.9	5942.4	363.19	26.08	1	3.28E-07 ***
lrt_2	247	5782.6	5920.6	337.11	23.682	1	1.14E-06 ***
lrt_3	246	5760.7	5901.3	313.42	29.44	1	5.77E-08 ***
lrt_4	245	5732.9	5876.2	283.98	0.66	1	0.4164
lrt_5	244	5734.2	5880.1	283.32	6.093	1	0.01357 *
lrt_6	243	5730.1	5878.6	277.23	4.781	1	0.02877 *
lrt_7	242	5727.3	5878.4	272.45	4.967	1	0.02584 *
lrt_8	241	5724.2	5878	267.48	2.839	1	0.09202 .
lrt_9	240	5723.4	5879.7	264.64	4.835	1	0.02788 *
lrt_10	239	5720.5	5879.4	259.81	4.944	1	0.02618 *
lrt_11	238	5717.5	5879	254.87	0.246	1	0.62023
lrt_12	237	5719.3	5883.4	254.62			

The chi-square test statistics and *p*-values are presented in Table A1.

Table 2. Monte Carlo Simulation Results for Asymptotic Properties

These codes replicate the simulation results reported in Table 2. First, we set up the covariance matrix to obtain the diagonal (ψ\psi), which is included in the population model. Note that running this code is not necessary because the values are already part of the population model. To change the sample size, modify sample.nobs= [# of sample size].

```
identity<-diag(1, 24,24)
phi<-matrix(c( 1, 0.3, 0.4,
               0.3, 1, 0.5,
               0.4, 0.5, 1
), 3,3)
ld<-matrix(c(0.65, 0.65, 0.7, 0.7, 0.7, 0.6, 0.6, 0.5,   0.6, 0, 0, 0, 0, 0, 0,0,  0, 0, 0, 0, 0, 0, 0, 0,
             0, 0, 0, 0, 0, 0.5, 0, 0,  0.6, 0.6, 0.6, 0.7, 0.7, 0.5, 0.5, 0.55, 0, 0, 0, 0.5, 0, 0, 0, 0,
             0, 0, 0, 0, 0, 0, 0, 0,  0, 0, 0, 0, 0, 0, 0, 0.55,  0.5, 0.5, 0.5, 0.5, 0.6, 0.6, 0.7, 0.7
), nrow=24, ncol=3)
ld2<-t(ld)
psi<-diag(1,24)-diag(ld %*% phi %*% ld2)
diag(psi)
```

```
popmodel<-'
f1 =~ 0.65*x1 + 0.65*x2 + 0.7*x3 + 0.7*x4 + 0.7*x5 + 0.7*x6 + 0.6*x7 + 0.5*x8 + 0.5*x9
f2 =~ 0.6*x9 + 0.6*x10 + 0.6*x11 + 0.7*x12 + 0.7*x13 + 0.5*x14 + 0.5*x15 + 0.65*x16 + 0.5*x6 +
0.75*x20
f3 =~ 0.5*x17 + 0.5*x18 + 0.5*x19 + 0.6*x20 + 0.6*x21 + 0.6*x22 + 0.7*x23 + 0.7*x24 + 0.45*x16
f3 ~~ 0.4*f1 + 0.5*f2
f1 ~~ 0.3*f2
f1 ~ 0*1
f2 ~ 0*1
f3 ~ 0*1
x1 ~~ 0.5775*x1
x2 ~~ 0.5775*x2
x3 ~~ 0.51*x3
x4 ~~ 0.51*x4
x5 ~~ 0.51*x5
x6 ~~ 0.05*x6
x7 ~~ 0.64*x7
x8 ~~ 0.21*x8
x9 ~~ 0.21*x9
x10 ~~ 0.64*x10
x11 ~~ 0.51*x11
x12 ~~ 0.51*x12
x13 ~~ 0.75*x13
x14 ~~ 0.75*x14
x15 ~~ 0.5775*x15
x16 ~~ 0.7975*x16
x17 ~~ 0.75*x17
x18 ~~ 0.75*x18
x19 ~~ 0.25*x19
x20 ~~ 0.64*x20
x21 ~~ 0.64*x21
x22 ~~ 0.64*x22
x23 ~~ 0.51*x23
x24 ~~ 0.51*x24
'
mymodel<-'
f1 =~ x1 + x2 + x3 + x4 + x5 + x6  + x7 + x8 + x9
f2 =~ x9 + x10 + x11 + x12 + x13 + x14 + x15 + x16 + x6 + x20
f3 =~ x17 + x18 + x19 + x20 + x21 + x22 + x23 + x24 + x16
f3 ~~ f1
f1 ~~ f2
'
```

```
##################################

    N=100

##################################
set.seed(100)
bin.1000 <- matrix(NA,nrow=1000, ncol=5)
for (i in 1:1000) {
 mydata<-simulateData(popmodel, model.type = "sem", meanstructure = FALSE,
sample.nobs=100L)
 fit1 <- cfa(model = mymodel, data=mydata, meanstructure=TRUE, int.ov.free = FALSE, std.lv =
FALSE, estimator = "ML")
 fitted(fit1)[1]
 pvalue <- lavInspect(fit1, "test")[[1]]$pvalue
 stat <- lavInspect(fit1, "test")[[1]]$stat
 nfi <- fitMeasures(fit1, "nfi")
 cfi <- fitMeasures(fit1, "cfi")
 rmsea<- fitMeasures(fit1, "rmsea")
 bin.1000[i,1]<-stat
 bin.1000[i,2]<-pvalue
 bin.1000[i,3]<-nfi
 bin.1000[i,4]<-cfi
 bin.1000[i,5]<-rmsea
}
T1 <- mean(bin.1000[, 1], na.rm = TRUE)
T1
sd(bin.1000[, 1], na.rm = TRUE)
p <- mean(bin.1000[, 2], na.rm = TRUE)
p
sd(bin.1000[, 1], na.rm = TRUE)
df<- lavInspect(fit1, "test")[[1]]$df
library(plyr)
x<-sum(bin.1000[,2]<0.05)
total<-length(bin.1000[,2])
rej_rate<-x/total
print(rej_rate)
quantile(bin.1000[, 3],.025)
quantile(bin.1000 [, 3],.975)
quantile(bin.1000[, 4],.025)
quantile(bin.1000 [, 4],.975)
quantile(bin.1000[, 5],.025)
quantile(bin.1000 [, 5],.975)
```