# Analysis of correlation matrix

*Mike Cheung*

*10 Apr 2019*

## Contents

## Fitting a correlation matrix as a covariance matrix

```
library(metaSEM)
```

```
## Loading required package: OpenMx

## To take full advantage of multiple cores, use:
##   mxOption(NULL, 'Number of Threads', parallel::detectCores()) #now
##   Sys.setenv(OMP_NUM_THREADS=parallel::detectCores()) #before library(OpenMx)

## "SLSQP" is set as the default optimizer in OpenMx.

## mxOption(NULL, "Gradient algorithm") is set at "central".

## mxOption(NULL, "Optimality tolerance") is set at "6.3e-14".

## mxOption(NULL, "Gradient iterations") is set at "2".
```

```
## Sample correlation matrix
my.cor <- Nohe15A1$data[[1]]
my.cor
```

```
##      W1   S1   W2   S2
## W1 1.00 0.29 0.58 0.22
## S1 0.29 1.00 0.24 0.57
## W2 0.58 0.24 1.00 0.27
## S2 0.22 0.57 0.27 1.00
```

```
## Sample size
my.n <- Nohe15A1$n[1]
my.n
```

```
## [1] 489
```

```
## Proposed model in lavaan syntax
model1 <- 'W2 ~ same*W1 + diff*S1
           S2 ~ diff*W1 + same*S1
           W1 ~~ w1WITHs1*S1
           W2 ~~ w2WITHs2*S2'
# plot(model1)

## Convert into RAM
RAM1 <- lavaan2RAM(model1, obs.variables=c("W1", "S1", "W2", "S2"))
RAM1
```

```
## $A
##    W1       S1       W2  S2
## W1 "0"      "0"      "0" "0"
## S1 "0"      "0"      "0" "0"
## W2 "0*same" "0*diff" "0" "0"
## S2 "0*diff" "0*same" "0" "0"
##
## $S
##    W1           S1           W2           S2
## W1 "0*W1WITHW1" "0*w1WITHs1" "0"          "0"
## S1 "0*w1WITHs1" "0*S1WITHS1" "0"          "0"
## W2 "0"          "0"          "0*W2WITHW2" "0*w2WITHs2"
## S2 "0"          "0"          "0*w2WITHs2" "0*S2WITHS2"
##
## $F
##    W1 S1 W2 S2
## W1  1  0  0  0
## S1  0  1  0  0
## W2  0  0  1  0
## S2  0  0  0  1
##
## $M
##   W1 S1 W2 S2
## 1  0  0  0  0
```

```r
my.data <- mxData(observed=my.cor, type="cov", numObs=my.n)

## Convert the matrices into mx matrices
A <- as.mxMatrix(RAM1$A, name="A")
S <- as.mxMatrix(RAM1$S, name="S")

## Use better starting values in S
diag(S$values) <- 0.5

F <- as.mxMatrix(RAM1$F, name="F")
exp  <- mxExpectationRAM("A","S","F",
                         dimnames=c("W1", "S1", "W2", "S2"))
funML  <- mxFitFunctionML()

## Fit it as a covariance matrix
model.cov <- mxModel("Cov matrix", my.data, A, S, F, exp, funML)

fit.cov <- mxRun(model.cov)
```

```
## Running Cov matrix with 8 parameters
```

```r
summary(fit.cov)
```

```
## Summary of Cov matrix
##
## free parameters:
##       name matrix row col   Estimate  Std.Error A
## 1     same      A  W2  W1 0.55501312 0.02662931
## 2     diff      A  S2  W1 0.06916124 0.02663246
## 3 W1WITHW1      S  W1  W1 0.99795501 0.06382214
## 4 w1WITHs1      S  S1  W1 0.28940696 0.04698845
## 5 S1WITHS1      S  S1  S1 0.99795504 0.06382214
## 6 W2WITHW2      S  W2  W2 0.65672797 0.04200332
## 7 w2WITHs2      S  S2  W2 0.10243804 0.03036496
## 8 S2WITHS2      S  S2  S2 0.67056639 0.04288843
##
## Model Statistics:
##                |  Parameters  |  Degrees of Freedom  |  Fit (-2lnL units)
##        Model:              8                      2               1498.209
##    Saturated:             10                      0               1498.039
## Independence:              4                      6               1952.000
## Number of observations/statistics: 489/10
##
## chi-square:  chisq ( df=2 ) = 0.1700241,  p = 0.9185012
## Information Criteria:
##       |  df Penalty  |  Parameters Penalty  |  Sample-Size Adjusted
## AIC:      -3.829976               16.17002                 16.47002
## BIC:     -12.214701               49.70892                 24.31715
## CFI: 1.004085
## TLI: 1.012255   (also known as NNFI)
## RMSEA: 0 *(Non-centrality parameter is negative)  [95% CI (0, 0.05021318)]
## Prob(RMSEA <= 0.05): 0.9747483
## timestamp: 2019-04-10 14:51:56
## Wall clock time: 0.109241 secs
## optimizer:  SLSQP
## OpenMx version number: 2.12.2
## Need help?  See help(mxSummary)
```

```
## The diagonals of the expected covariance matrix (expCov) are close to ones but not exactly ones.
fit.cov$output$algebras
```

```
## $`Cov matrix.fitfunction`
##          [,1]
## [1,] 1498.209
## attr(,"expCov")
##           [,1]      [,2]      [,3]      [,4]
## [1,] 0.9979550 0.2894070 0.5738939 0.2296445
## [2,] 0.2894070 0.9979550 0.2296445 0.5738939
## [3,] 0.5738939 0.2296445 0.9911291 0.2695850
## [4,] 0.2296445 0.5738939 0.2695850 1.0049675
## attr(,"expMean")
## <0 x 0 matrix>
## attr(,"SaturatedLikelihood")
## [1] 1498.039
## attr(,"IndependenceLikelihood")
## [1] 1952
```

## Fitting a correlation matrix as a correlation matrix by ensuring that the diagonals are ones.

```
## No moderator ## Proposed model in lavaan syntax
model2 <- 'W2 ~ same*W1 + diff*S1
```

```
           S2 ~ diff*W1 + same*S1
           W1 ~~ w1WITHs1*S1
           W2 ~~ w2WITHs2*S2
           W1 ~~ 1*W1   # Fix the variance of the independent variables at 1
           S1 ~~ 1*S1'
# plot(model2)

## Convert into RAM
RAM2 <- lavaan2RAM(model2, obs.variables=c("W1", "S1", "W2", "S2"))
RAM2
```

```
## $A
##    W1       S1       W2  S2
## W1 "0"      "0"      "0" "0"
## S1 "0"      "0"      "0" "0"
## W2 "0*same" "0*diff" "0" "0"
## S2 "0*diff" "0*same" "0" "0"
##
## $S
##    W1           S1           W2           S2
## W1 "1"          "0*w1WITHs1" "0"          "0"
## S1 "0*w1WITHs1" "1"          "0"          "0"
## W2 "0"          "0"          "0*W2WITHW2" "0*w2WITHs2"
## S2 "0"          "0"          "0*w2WITHs2" "0*S2WITHS2"
##
## $F
##    W1 S1 W2 S2
## W1  1  0  0  0
## S1  0  1  0  0
## W2  0  0  1  0
## S2  0  0  0  1
##
## $M
##   W1 S1 W2 S2
## 1  0  0  0  0
```

```
## Create a model-implied correlation matrix "impliedR"
## M0 includes several intermediate mx matrices required to calculate the "impliedR".
M0 <- create.vechsR(A0=RAM2$A, S0=RAM2$S, F0=RAM2$F)

## Create a template of the model
model.cor <- mxModel("Cor matrix", my.data, funML)

## Add the matrices in M0 to the template
for (i in seq_along(M0)) {
  model.cor <- mxModel(model.cor, M0[[i]])
}

## Specify the model implied correlation matrix
model.cor <- mxModel(model.cor,
                     mxExpectationNormal(covariance='impliedR',
                                         dimnames=c("W1", "S1", "W2", "S2")))

fit.cor <- mxRun(model.cor)
```

```
## Running Cor matrix with 4 parameters
```

```
## The df is incorrect as it has not been adjusted yet.
summary(fit.cor)
```

```
## Summary of Cor matrix
## 
## free parameters:
##       name matrix row col   Estimate  Std.Error A
## 1     same     A0  W2  W1 0.55550421 0.02000930
## 2     diff     A0  S2  W1 0.06920345 0.02621189
## 3 w1WITHs1     S0  S1  W1 0.29058663 0.03972111
## 4 w2WITHs2     S0  S2  W2 0.10263036 0.02974706
## 
## Model Statistics: 
##                |  Parameters  |  Degrees of Freedom  |  Fit (-2lnL units)
##        Model:              4                      6              1498.267
##    Saturated:             10                      0              1498.039
## Independence:              4                      6              1952.000
## Number of observations/statistics: 489/10
## 
## chi-square:  chisq ( df=6 ) = 0.2272597,  p = 0.9997754
## Information Criteria: 
##      |  df Penalty  |  Parameters Penalty  |  Sample-Size Adjusted
## AIC:      -11.77274                8.22726                 8.309904
## BIC:      -36.92692               24.99671                12.300821
## CFI: 1.012887 
## TLI: 1.012887   (also known as NNFI) 
## RMSEA: 0 *(Non-centrality parameter is negative)  [95% CI (0, 0)]
## Prob(RMSEA <= 0.05): 0.9999936
## timestamp: 2019-04-10 14:51:56 
## Wall clock time: 0.08937621 secs 
## optimizer:  SLSQP 
## OpenMx version number: 2.12.2 
## Need help?  See help(mxSummary)
```

```
## Adjust the df manually
summary(fit.cor, SaturatedDoF=4)
```

```
## Summary of Cor matrix
## 
## free parameters:
##       name matrix row col   Estimate  Std.Error A
## 1     same     A0  W2  W1 0.55550421 0.02000930
## 2     diff     A0  S2  W1 0.06920345 0.02621189
## 3 w1WITHs1     S0  S1  W1 0.29058663 0.03972111
## 4 w2WITHs2     S0  S2  W2 0.10263036 0.02974706
## 
## Model Statistics: 
##                |  Parameters  |  Degrees of Freedom  |  Fit (-2lnL units)
##        Model:              4                      6              1498.267
##    Saturated:              6                      4              1498.039
## Independence:              4                      6              1952.000
## Number of observations/statistics: 489/10
## 
## chi-square:  chisq ( df=2 ) = 0.2272597,  p = 0.8925883
## Information Criteria:
##       |  df Penalty  |  Parameters Penalty  |  Sample-Size Adjusted
## AIC:      -11.77274                8.22726                 8.309904
## BIC:      -36.92692               24.99671                12.300821
## CFI: 1.003922
## TLI: 1.003922   (also known as NNFI)
## RMSEA: 0 *(Non-centrality parameter is negative)  [95% CI (0, 0.05616586)]
## Prob(RMSEA <= 0.05): 0.9661626
## timestamp: 2019-04-10 14:51:56
## Wall clock time: 0.08937621 secs
## optimizer:  SLSQP
## OpenMx version number: 2.12.2
## Need help?  See help(mxSummary)
```

```
## The diagonals of the expected covariance matrix (expCov) are exactly ones.
# mxEval(impliedR, fit.cor)
fit.cor$output$algebras
```

```
## $`Cor matrix.Amatrix`
##             [,1]       [,2] [,3] [,4]
## [1,] 0.00000000 0.00000000    0    0
## [2,] 0.00000000 0.00000000    0    0
## [3,] 0.55550421 0.06920345    0    0
## [4,] 0.06920345 0.55550421    0    0
##
## $`Cor matrix.Smatrix`
##            [,1]      [,2]      [,3]      [,4]
## [1,] 1.0000000 0.2905866 0.0000000 0.0000000
## [2,] 0.2905866 1.0000000 0.0000000 0.0000000
## [3,] 0.0000000 0.0000000 0.6642840 0.1026304
## [4,] 0.0000000 0.0000000 0.1026304 0.6642840
##
## $`Cor matrix.impliedR`
##            [,1]      [,2]      [,3]      [,4]
## [1,] 1.0000000 0.2905866 0.5756138 0.2306255
## [2,] 0.2905866 1.0000000 0.2306255 0.5756138
## [3,] 0.5756138 0.2306255 1.0000000 0.2705783
## [4,] 0.2306255 0.5756138 0.2705783 1.0000000
##
## $`Cor matrix.vechsR`
##            [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
## [1,] 0.2905866 0.5756138 0.2306255 0.2306255 0.5756138 0.2705783
##
## $`Cor matrix.SnoErr`
##            [,1]      [,2]      [,3]      [,4]
## [1,] 1.0000000 0.2905866 0.0000000 0.0000000
## [2,] 0.2905866 1.0000000 0.0000000 0.0000000
## [3,] 0.0000000 0.0000000 0.0000000 0.1026304
## [4,] 0.0000000 0.0000000 0.1026304 0.0000000
##
## $`Cor matrix.invS0`
##            [,1]      [,2]      [,3]      [,4]
## [1,] 1.0000000 0.2905866 0.5756138 0.2306255
## [2,] 0.2905866 1.0000000 0.2306255 0.5756138
## [3,] 0.5756138 0.2306255 0.3357160 0.2705783
## [4,] 0.2306255 0.5756138 0.2705783 0.3357160
##
## $`Cor matrix.invS1`
##           [,1]      [,2]      [,3]      [,4]
## [1,] 1.0000000 0.2905866 0.5756138 0.2306255
## [2,] 0.2905866 1.0000000 0.2306255 0.5756138
## [3,] 0.5756138 0.2306255 1.0000000 0.2705783
## [4,] 0.2306255 0.5756138 0.2705783 1.0000000
##
## $`Cor matrix.E1`
##      [,1] [,2]     [,3]     [,4]
## [1,]    0    0 0.000000 0.000000
## [2,]    0    0 0.000000 0.000000
## [3,]    0    0 0.664284 0.000000
## [4,]    0    0 0.000000 0.664284
##
## $`Cor matrix.fitfunction`
##          [,1]
## [1,] 1498.267
## attr(,"expCov")
##           [,1]      [,2]      [,3]      [,4]
## [1,] 1.0000000 0.2905866 0.5756138 0.2306255
## [2,] 0.2905866 1.0000000 0.2306255 0.5756138
## [3,] 0.5756138 0.2306255 1.0000000 0.2705783
## [4,] 0.2306255 0.5756138 0.2705783 1.0000000
## attr(,"expMean")
## <0 x 0 matrix>
## attr(,"SaturatedLikelihood")
## [1] 1498.039
## attr(,"IndependenceLikelihood")
## [1] 1952
```