Supplementary Material of "A Comparison of Bayesian and Frequentist Model Selection Methods for Factor Analysis Models"

Zhao-Hua Lu, Sy-Miin Chow, Eric Loken

Department of Human Development and Family Studies, Pennsylvania State University, University Park, Pennsylvania

Author Note

Zhao-Hua Lu, Department of Human Development and Family Studies, Pennsylvania State University, University Park, Pennsylvania; Sy-Miin Chow, Department of Human Development and Family Studies, Pennsylvania State University, University Park, Pennsylvania; Eric Loken, Department of Human Development and Family Studies, Pennsylvania State University, University Park, Pennsylvania.

Funding for this study was provided by NSF grant BCS-0826844, and NIH grant R01GM105004. Correspondence concerning this article can be addressed to Zhao-Hua Lu, the Pennsylvania State University, 413A Biobehavioral Health Building, University Park, PA 16802 or by email to zhaohua.lu@gmail.com.

Supplementary Material of "A Comparison of Bayesian and Frequentist Model Selection Methods for Factor Analysis Models"

The main scripts are as follows. First, we loaded some R packages, sourced some R functions, and defined some constants of dimensions. We simulated data from factor analysis model with Simulation.R, which can be changed for different settings. Then we inputted data, defined hyperparameters, identification conditions, and initial values of parameters, which were stored as lists to be used by the function for Bayesian factor analysis. The MCMC sampling was handled by the function BFA. The MCMC samples were stored in a list, which was used to calculate the Bayesian MCC and posterior model probabilities based on SSP.

```
# Load some libraries and R functions
install.packages("mvtnorm")
install.packages("loo")


library(stats)
library(MASS)
library(mvtnorm)
library(loo)


source("Routine.R",local=T)
source("BFA.R")
source("MCCRoutine.R")

# Dimension Setting
NMCMC <- 8000  # Number of MCMC samples generated
NBurn <- 4000  # Number of MCMC samples discarded as burn-in samples
```

```
Nthin <- 1     # thinning of MCMC sample, the number of MCMC
   iterations is NMCMC * Nthin
N <- 100 # Number of subjects
NY <- 9  # Number of items
NK <- 3  # Number of factors
NZ <- NK

# Simulate data

set.seed(1000)

source("SimulateData.R")


#set.seed(1000)

# Input data
DataMat<-read.table("SimulatedData.txt",header=F)
# Define the hyperparameters of  prior distributions
source("Prior.R",local=T)
# Define the identification conditions
source("ind.R",local=T)
# Define the initial values of parameters and latent factors
source("init1.R",local=T)

#set.seed(10)
# Generate MCMC samples for the factor analysis model
# The results is a list containing MCMC samples of parameters and
   latent factors
```

```
46 BFAResult <- BFA(DataMat,InitialValues,Identification,Hyperpara,
      NMCMC=NMCMC,NBurn=NBurn,Nthin=Nthin,StoreLatentVariables = T)
47
48 #dump(c("BFAResult"),file="MCMCSamples.R")
49 save("BFAResult",file="MCMCSamples.RData")
50
51 # Calculate posterior mean, SE and posterior inclusion probabilities
       of the parameters stored in a list
52 MCMCSummary(BFAResult)
53
54 #set.seed(10)
55 # Calculate the model comparison statistics
56 MCCResults <- BayesianMCC(BFAResult,DataMat,Hyperpara,Identification
      ,BridgeSamplerMargLik=T,BIC=T,DIC=T,LOO=T)
57
58
59 # Only use if SSP is used
60 # Store the structure of the candidate models in a list
      CandidateModels
61 source("CandidateModelsIdentification.R")
62 # Calculate the estimated marginal model probability by SSP
63 MMP <- MMP_SSP(BFAResult$PIP,CandidateModels)
```

A. MCMC Sampling

The full conditional distributions of the parameters and factors used in the MCMC sampling may be found in the literature (Lee, 2007; Lu, Chow, & Loken, 2016; Press, 2002) and other papers in this issue. We outline the key R codes used to implement the sampling

of these conditional distributions. The complete code will be posted on https://github.com/zhaohualu/BSEM_SSP. We refer the readers to Lee (2007) for the derivation and mathematical expression of these distributions. We will directly refer to the equations in Lee (2007) with prefix L.

Sampling the factors

The factors can be generated from their full conditional distribution which is a multivariate normal distribution. The derivation can be found in equation (L4.24) on p.

Alternatively, we can sample the factors using Metropolis-Hastings (MH) algorithm. We chose this approach as it leads to faster R code implementation. The derivation can be found in Equations (L8.11) - (L8.13).

1. First, we calculated the covariance matrix (L8.12) of the proposal distribution in the MH algorithm. This is a simplified version compared to (L8.12) as there is no endogenous latent variable (dependent latent variable in SEM) in the factor analysis model.

```
ISG <- crossprod(inv.sqrt.PSX * LY)
if (NZ > 0){
ISG[(NM + 1):NK, (NM + 1):NK] <- ISG[(NM + 1):NK, (NM + 1):NK
   ] + inv.PHI
}
ISG <- ISG
SIG <- chol2inv(chol(ISG))
```

2. Then we generate proposal samples (OMEN) for the factors by add a random pertubation to the current values of the factors (Omega).

```
cSIG<-chol(SIG)
```

```
2   OMEN<-Omega+crossprod(cSIG,matrix(rnorm(N*NK,0,1),nrow=NK,
       ncol=N))
```

3. The proposal sample is accepted as the new values of the factors with probability equal to $exp(-0.5(r2 - r1))$, where r_2 and r_1 are the log likelihood values of the existing and proposal values of the factor.

```
1   r1 <- log.likelihood(Y - Ycen, LY, Omega, inv.sqrt.PSX,
2   c.inv.PHI)
3   r2 <- log.likelihood(Y - Ycen, LY, OMEN, inv.sqrt.PSX,
4   c.inv.PHI)
5   r <- exp(0.5 * (r1 - r2))
6   comr <- runif(N)
7   crit <- (comr < r)
8   COV[(2 + NANA):(1 + NANA + NK), crit] <- Omega[, crit] <-
       OMEN[, crit]
```

Sampling $\boldsymbol{\mu}$, $\boldsymbol{\Lambda}$, $\boldsymbol{\Psi}$ and $\mathbf{R}$

Given the prior distributions of $\boldsymbol{\Lambda}$, $\boldsymbol{\Lambda}$, and $\boldsymbol{\Psi}$, they are sampled together. The samples are generated from the full conditional distributions in pp. 83 - 85 of Lee (2007).

In case SSP is assigned to certain elements in the loading matrix, the corresponding elements in $\mathbf{R}$ is sampled from the full conditional distributions are derived in Lu et al. (2016).

```
1 # Sampleing the indicator in SSP, intercepts, loading matrix and
2 # unique variance for every item
3
```

```
4  XX <- tcrossprod(COV)
5
6  for (j in 1:NY) {
7
8  # Sampling the R for SSP in case some elements assigned SSP
9
10 subs <- ICE[j, ] == 0  # fixed item
11 Ycen <- as.vector(Y[j, , drop = FALSE] - COE[j,
12 (subs), drop = F] %*% COV[(subs), , drop = F])
13 XY <- COV %*% Ycen
14 Ycens2 <- sum(Ycen^2)
15
16 iPSigb0 <- diag(PPCOE[j, ])
17 Pmean <- PCOE[j, ]
18
19 idx0 <- IDE[j, ] > 0  # current !=0 components
20
21 idxtmp <- which(ICE[j, ] == 2)
22 for (k in idxtmp) {
23 if (IDE[j, k] > 0) {
24 idx1 <- idx0
25 idx2 <- idx0
26 idx2[k] <- F
27 } else {
28 idx1 <- idx0
29 idx2 <- idx0
30 idx1[k] <- T
31 }
32
```

```
iPSigb1 <- iPSigb0[idx1, idx1, drop = F]
Pb1 <- Pmean[idx1]
iSigb1 <- XX[idx1, idx1] + iPSigb1
Sigb1 <- solve(iSigb1)
bets1 <- Sigb1 %*% (XY[idx1] + iPSigb1 %*% Pb1)

# Calcualte the posterior proability of the indicators

logR <- log(pr[j, k]/(1 - pr[j, k]))

if (sum(idx2) > 0) {
iPSigb2 <- iPSigb0[idx2, idx2, drop = F]
Pb2 <- Pmean[idx2]
iSigb2 <- XX[idx2, idx2] + iPSigb2
Sigb2 <- solve(iSigb2)
bets2 <- Sigb2 %*% (XY[idx2] + iPSigb2 %*% Pb2)

logR <- logR - log(sqrt(det(Sigb2)))  #*det(iPSigb2)
logR <- logR + (N/2 + alphax[j]) * log(Ycens2 +
sum(Pb2 * (iPSigb2 %*% Pb2)) - sum(bets2 *
(iSigb2 %*% bets2)))
}


logR <- logR - (N/2 + alphax[j]) * log(Ycens2 +
sum(Pb1 * (iPSigb1 %*% Pb1)) - sum(bets1 * (iSigb1 %*%
bets1)))
logR <- logR + log(sqrt(det(Sigb1)))  #det(iPSigb1)
if (PPCOE[j, k] > 0)
```

```
logR <- logR + log(sqrt(PPCOE[j, k]))
R <- exp(logR)
p <- 1 - 1/(1 + R)
itmp <- rbinom(1, 1, p)
IDE[j, k] <- itmp
idx0 <- IDE[j, ] > 0  # current !=0 components
}
idx1 <- IDE[j, ] == 0 & ICE[j, ] == 2
COE[j, idx1] <- 0

# Sampling the intercepts, loadings and unique variances

subs <- IDE[j, ] > 0  # current active components to be optimized.
Ycen <- COE[j, (!subs), drop = F] %*% COV[(!subs),
, drop = F]

omesub <- COV[subs, , drop = FALSE]
ssubs <- sum(subs)
iPSiginv <- diag(PPCOE[j, subs], ssubs)
Pmean1 <- PCOE[j, subs]

# Sampling unique variances
Ycen <- as.vector(Y[j, , drop = FALSE] - Ycen)
alphastar <- alphax[j] + N/2
betastar <- betax[j] + 1/2 * sum(Ycen^2)
if (ssubs > 0) {
calsmnpsx <- chol2inv(chol((XX[subs, subs, drop = F] +
iPSiginv)))
temp <- (omesub %*% Ycen + iPSiginv %*% Pmean1)
```

```
LYnpsx <- calsmnpsx %*% temp
betastar <- betastar + 1/2 * (sum(Pmean1 * iPSiginv *
Pmean1) - sum(temp * LYnpsx))
}
iPSX[j] <- rgamma(1, shape = alphastar, rate = betastar)
PSX[j] <- 1/iPSX[j]
inv.sqrt.PSX[j] <- sqrt(iPSX[j])

# Sampling the intercepts and loadings
if (ssubs > 0) {
COE[j, idx0] <- mvrnorm(1, LYnpsx, PSX[j] * calsmnpsx)
}
}
```

Sampling Φ

The covariance matrix is generated from Inverse-Wishart distribution (L4.30).

```
inv.PHI[, ] <- rWishart(1, RouZero + N, solve(tcrossprod(Omega[(NM +
1):NK, , drop = F]) + IWMat))
c.inv.PHI <- chol(inv.PHI)
PHI <- chol2inv(c.inv.PHI)
```

B. Calculation of MCC

The calculation is handled by the function BayesianMCC. Some code in the function are listed below.

B.1. Calculate the marginal probability for Bayes Factor

```
#### LOG LIKELIHOOD BASED ON MCMC SAMPLES (mlfytk)
Y1 <- DataMat
mlfytk <- numeric(nrow(mcmc.lam))
for (j in 1:nrow(mcmc.lam)) {
Lambj <- matrix(mcmc.lam[j, ], nrow = NY, ncol = NK,
byrow = T)
PHIj <- matrix(mcmc.phx[j, ], nrow = NZ, ncol = NZ,
byrow = T)
Sigmaj <- diag(mcmc.sigma[j, ])

mlfytk[j] <- LFYTK(Y1, mcmc.mu[j, ], Lambj, Sigmaj,
PHIj)
}

#### LOG PRIOR DENSITY FOR MCMC SAMPLES (SLPT)
tPLY <- t(PLY)
tPPLY <- t(PPLY)
# Prior density of Intercept
SLPT0 <- apply(mcmc.mu, FUN = LPT1, MAR = 1, MLam = PMU,
VLam = 1/(PPMU))
# Prior density of free elements in the loading matrix
SLPT1 <- apply(mcmc.lam[, lamfreeidx], FUN = LPT1, MAR = 1,
MLam = tPLY[lamfreeidx], VLam = 1/(tPPLY[lamfreeidx]))

SLPT2 <- numeric(Nrec)
SLPT3 <- numeric(Nrec)
for (j in 1:Nrec) {
```

```
28 # Prior density of unique variance
29 SLPT2[j] <- LPT2(mcmc.sigma[j, ], a = alphax, b = betax)
30 # Prior density of factor covariance matrix
31 PHItmp <- matrix(mcmc.phx[j, ], NZ, NZ, byrow = F)
32 SLPT3[j] <- diwish(PHItmp, RouZero, IWMat, logflag = T)
33 }
34 SLPT <- SLPT0 + SLPT1 + SLPT2 + SLPT3
35
36
37
38 ### Calculate the parameters of the g distibutions
39
40 # Mean and Covariance of multivariate normal distribution for
41 # the Intercepts
42 mmu <- colMeans(mcmc.mu)
43 vmu <- cov(mcmc.mu)
44
45 # Mean and Covariance of multivariate normal distribution for
46 # the free elements in the loading matrix
47 mLam <- colMeans(mcmc.lam[, lamfreeidx])
48 vLam <- cov(mcmc.lam[, lamfreeidx])
49
50 # Shape and rate parameters of inverse gamma distribution for
51 # the unique variances
52 pm <- colMeans(mcmc.sigma)
53 pv <- apply(mcmc.sigma, FUN = var, MAR = 2)
54 pa <- as.numeric(pm^2/pv + 2)
55 pb <- as.numeric((pa - 1) * pm)
56
```

```
# Parameters of the inverse Wishart distribution for the factor
# covariance matrix
Exiv <- colMeans(mcmc.omega)
Exi <- matrix(Exiv, nrow = NK, ncol = N)
Exi2 <- tcrossprod(Exi) + IWMat
Erho <- mean((Exi2)/Ephx) + NZ + 1
if (Erho <= 0) {
Erho <- mean(diag(Exi2)/diag(Ephx)) + NZ + 1
}

## Denominator

### LOG DENSITY FUNCTION of proposal g distributions OF MCMC
### SAMPLES (Sgf)

Sgf0 <- gf1(mcmc.mu, mmu, vmu)
Sgf1 <- gf1(mcmc.lam[, lamfreeidx], mLam, vLam)
Sgf2 <- apply(mcmc.sigma, FUN = gf2, MAR = 1, pa, pb)
Sgf3 <- numeric(Nrec)
for (j in 1:Nrec) {
PHItmp <- matrix(mcmc.phx[j, ], NZ, NZ, byrow = F)
Sgf3[j] <- diwish(PHItmp, Erho, Exi2, logflag = T)
}
Sgf <- Sgf0 + Sgf1 + Sgf2 + Sgf3



#### GENERATE ADDITIONAL SAMPLES from the g density FOR BRIDGE
```

```
#### SAMPLER

gpsx <- matrix(1/rgamma(Nrec * length(pa), rep(pa, each = Nrec),
rate = rep(pb, each = Nrec)), nrow = Nrec, ncol = length(pm))
gmu <- mvrnorm(Nrec, mmu, vmu)
gLam <- mvrnorm(Nrec, mLam, vLam)
giPHI <- rWishart(Nrec, Erho, solve(Exi2))
gPHI <- array(0, dim = dim(giPHI))
for (j in 1:Nrec) {
gPHI[, , j] <- solve(giPHI[, , j])
}

#### Log LIKELIHOOD BASED ON ADDITIONAL (g) SAMPLES (gmlfytk)
gmlfytk <- numeric(Nrec)
for (j in 1:Nrec) {
tLambj <- matrix(mcmc.lam[1, ], nrow = NK, ncol = NY)
# tLambj[lamfixidx] = lamfixval
tLambj[lamfreeidx] <- gLam[j, ]
Lambj <- t(tLambj)
PHIj <- gPHI[, , j]
Sigmaj <- diag(gpsx[j, ])

gmlfytk[j] <- LFYTK(Y1, gmu[j, ], Lambj, Sigmaj, PHIj)
}

#### Log PRIOR FOR ADDITIONAL (g) SAMPLES (SLPT)
tPLY <- t(PLY)
tPPLY <- t(PPLY)
# Prior density of Intercept
```

```
gSLPT0 <- apply(gmu, FUN = LPT1, MAR = 1, MLam = PMU, VLam = 1/(PPMU
   ))
# Prior density of free elements in the loading matrix
gSLPT1 <- apply(gLam, FUN = LPT1, MAR = 1, MLam = tPLY[lamfreeidx],
VLam = 1/(tPPLY[lamfreeidx]))
gSLPT2 <- numeric(Nrec)
gSLPT3 <- numeric(Nrec)

for (j in 1:Nrec) {
# Prior density of unique variance
gSLPT2[j] <- LPT2(gpsx[j, ], a = alphax, b = betax)
# Prior density of factor covariance matrix
PHItmp <- gPHI[, , j]
gSLPT3[j] <- diwish(PHItmp, RouZero, IWMat, logflag = T)
}
gSLPT <- gSLPT0 + gSLPT1 + gSLPT2 + gSLPT3


# g LOG DENSITY OF g SAMPLES (Sgf) Intercept
gSgf0 <- gf1(gmu, mmu, vmu)
# Free loading elements
gSgf1 <- gf1(gLam, mLam, vLam)
# Unique Variance
gSgf2 <- apply(gpsx, FUN = gf2, MAR = 1, pa, pb)
# Factor Covariance
gSgf3 <- numeric(Nrec)
for (j in 1:Nrec) {
PHItmp <- gPHI[, , j]
gSgf3[j] <- diwish(PHItmp, Erho, Exi2, logflag = T)
```

```
}
#
gSgf <- gSgf0 + gSgf1 + gSgf2 + gSgf3

# Bridge Sampler
lN <- 0.5 * (gmlfytk + gSLPT - gSgf)
lD <- -0.5 * (mlfytk + SLPT - Sgf)
mlN <- mean(lN)
mlD <- mean(lD)
Marglik_BS <- log(mean(exp(lN - mlN))/mean(exp(lD - mlD))) +
(mlN - mlD)
```

B.2. Calculate BIC

```
Emu <- colMeans(mcmc.mu)
ELam <- matrix(colMeans(mcmc.lam), nrow = NY, ncol = NK,
byrow = T)
Esigma <- colMeans(mcmc.sigma)
Ephx <- matrix(colMeans(mcmc.phx), nrow = NZ, ncol = NZ,
byrow = T)

library(mvtnorm)
# BIC, log likelihood and number of parameters
BIC <- -2 * sum(dmvnorm(DataMat, Emu, ELam %*% Ephx %*%
t(ELam) + diag(Esigma), log = T)) + (length(lamfreeidx) +
NY + NY + NZ * (NZ + 1)/2) * log(N)
```

B.3. Calculate DIC

```
# Log likelihood at posterior mean
dbt <- -2 * sum(dmvnorm(DataMat, Emu, ELam %*% Ephx %*%
t(ELam) + diag(Esigma), log = T))
dtb <- 0
# Posterior mean of log likelihood
for (j in 1:Nrec) {
Lambj <- matrix(mcmc.lam[j, ], nrow = NY, ncol = NK,
byrow = T)
PHIj <- matrix(mcmc.phx[j, ], nrow = NZ, ncol = NZ,
byrow = T)
Sigmaj <- diag(mcmc.sigma[j, ])

dtb <- dtb + (-2 * sum(dmvnorm(DataMat, mcmc.mu[j, ],
Lambj %*% PHIj %*% t(Lambj) + (Sigmaj), log = T)))
}
dtb <- dtb/Nrec
pd <- dtb - dbt
DIC <- dtb + 2 * pd
```

B.4. Calculate LOO-PSIS

```
## Calculate LOO with the the loo function in the loo package
## Create a matrix recording the log likelihood of each subject
## givenparameters in each MCMC iteration , # of row is number of
## MCMC sample , # of column if # of subject
LogLikMcMat <- matrix(nrow = Nrec, ncol = nrow(DataMat))
for (j in 1:Nrec) {
```

```
Lambj <- matrix(mcmc.lam[j, ], nrow = NY, ncol = NK,
byrow = T)
PHIj <- matrix(mcmc.phx[j, ], nrow = NZ, ncol = NZ,
byrow = T)
Sigmaj <- diag(mcmc.sigma[j, ])

LogLikMcMat[j, ] <- dmvnorm(DataMat, mcmc.mu[j, ], Lambj %*%
PHIj %*% t(Lambj) + (Sigmaj), log = T)
}
lootmp <- loo(LogLikMcMat)
LooEst <- lootmp$looic
```

B.5. Calculate posterior model probability with SSP

```
MMP_SSP <- function(PIP, CandidateModels) {

# A vector record the estimated marginal model probability by
# SSP
TrueModPostProb <- numeric(length(CandidateModels))
names(TrueModPostProb) <- names(CandidateModels)
for (mcrep in 1:dim(PIP)[1]) {
EILYtmp <- PIP[mcrep, , ]

# Count the number of iteration where the candidate model is the
# selected model
for (j in 1:length(CandidateModels)) {
# Index of free parameters to be compared
Tidx <- CandidateModels[[j]] == 3
```

```
# Index of fixed parameters to be compared
Fidx <- CandidateModels[[j]] == 2
# Count
TrueModPostProb[j] <- TrueModPostProb[j] + (all(EILYtmp[Tidx] ==
1) & all(EILYtmp[Fidx] == 0))
}
}
print("Marginal model probability calculated by SSP")
print(TrueModPostProb)
return(TrueModPostProb)
}
```

The point mass function in the SSP requires the use of specific Gibbs sampler (Liu, 1994) to generate MCMC samples. We implemented the collapsed Gibbs sampler (Liu, 1994) in R. The R code is available at [website address to be provided upon the acceptance of this manuscript]. A variant of the SSP is the hierarchical normal mixture prior (George & McCulloch, 1993) which replaces the point mass function with a normal distribution with a very small variance. This prior distribution can be implemented in JAGS through the Gibbs sampler. However, it introduces an additional tuning parameter (the small variance) that affects its performance and slows down the convergence of the MCMC algorithms compared to the SSP approach we used (O'Hara & Sillanpää, 2009).

C. Definitions of Some Bayesian MCC

BIC

The BIC is expressed as:

$$BIC = -2\log p_s(\mathbf{Y}|\bar{\boldsymbol{\theta}}_s) + (2p + ||\boldsymbol{\Lambda}_s||_0 + q(q+1)/2)log(n), \quad \text{(S1)}$$

where $||\mathbf{\Lambda}_s||_0$ is the number of parameters in $\mathbf{\Lambda}_s$ in this particular context. Just as the BIC is often used in the frequentist context for model selection, the first term in (S1) measures goodness-of-fit and the second term characterizes the model complexity in terms of the number of parameters in the model (1). Practically, we calculate the estimate of BIC by replacing the posterior means in (S1) with the sample means of the posterior samples from the MCMC algorithm. Sample code for calculating the BIC is given in Section B.2 in the supplementary material.

DIC

DIC is computed as

$$DIC \quad = \quad -2\log p_s(\mathbf{Y}|\bar{\boldsymbol{\theta}}_s) + 2p_D, \tag{S2}$$

where the first term is the same as BIC, and the second term

$$p_D = E_{\boldsymbol{\theta}_s|\mathbf{Y}}[-2\log p_s(\mathbf{Y}|\boldsymbol{\theta}_s)] + 2\log p_s(\mathbf{Y}|\bar{\boldsymbol{\theta}}_s)$$

is a measure of model complexity known as the effective number of parameters. The expectation in the first term is taken of the deviance, $-2\log p_s(\mathbf{Y}|\boldsymbol{\theta}_s)$, with respect to $p_s(\boldsymbol{\theta}_s|\mathbf{Y})$, and may be approximated by

$$\frac{1}{N_1}\sum_{t=1}^{N_1} -2\log p_s(\mathbf{Y}|\boldsymbol{\theta}_s^{(t)}), \tag{S3}$$

where $\boldsymbol{\theta}_s^{(t)}$ is the tth MCMC sample from $p_s(\boldsymbol{\theta}_s|\mathbf{Y})$ obtained during estimation N_1 represents the total number of MCMC samples generated.

Despite the DIC's practical advantages, its theoretical justification is relatively weak and it has known limitations in some modeling scenarios. For instance, Celeux, Forbes, Robert, and Titterington (2006) showed that the DIC is problematic in mixture models and compared several modified DIC for models with missing data and latent variables. For latent variable models where the closed form of the observed likelihood, $p_s(\mathbf{Y}|\boldsymbol{\theta}_s^{(t)})$, is not available, the DIC provided by software such as OpenBUGS and JAGS is usually

problematic as shown in Celeux et al. (2006). For the CFA models considered in the present context, if a closed form of the likelihood is not available as in Equation (3), then the MCMC algorithm used for estimation purposes is usually based on the augmented posterior distribution, $p_j(\boldsymbol{\Omega}, \boldsymbol{\theta}_j^{(t)}|\mathbf{Y})$, where the parameters and latent factors are treated equally and both have to be integrated out – usually via MCMC approximations – to obtain the likelihood function. In these cases, the resulting DIC is different from that in (S2) and was shown to be less satisfactory. Celeux et al. (2006) derived other versions of DIC which can not be computed directly in standard software. Here, because our model of interest does have a likelihood expression in closed form, we calculated the DIC (S2) using Equation (3) directly with our own R script (see Section B.3 of the supplementary material) rather than using the DIC output from JAGS.

In addition, the DIC tends to select over-fitted models. The reason is that in standard use of the DIC, the same observed data are used to estimate the parameters and evaluate the model in (S2). Ideally, a set of replicated data $\mathbf{Y}^{rep}$ should be used. Modifications of the DIC have been proposed to address this problem (see, for example, Ando, 2010), but often entail complicated posterior computations and are thus not pursued routinely in empirical practice (for a thorough review see Spiegelhalter, Best, Carlin, & van der Linde, 2014).

References

Ando, T. (2010). *Bayesian Model Selection and Statistical Modeling*. Boca Raton, Florida: Chapman and Hall/CRC.

Celeux, G., Forbes, F., Robert, C. P., & Titterington, D. M. (2006). Deviance information criteria for missing data models. *Bayesian Analysis*, *1*(4), 651–673.

George, E. I. & McCulloch, R. E. (1993). Variable Selection via Gibbs Sampling. *Journal of the American Statistical Association*, *88*, 881–889. doi:10.1080/01621459.1993.10476353

Lee, S.-Y. (2007). *Structural equation modeling: A Bayesian approach*. The Atrium, Southern Gate, Chichester, West Sussex PO19 8SQ, England: John Wiley & Sons.

Liu, J. S. (1994). The Collapsed Gibbs Sampler in Bayesian Computations with Applications to a Gene Regulation Problem. *Journal of the American Statistical Association*, *89*, 958–966. doi:10.1080/01621459.1994.10476829

Lu, Z.-H., Chow, S.-M., & Loken, E. (2016). Bayesian factor analysis as a variable-selection problem: alternative priors and consequences. *Multivariate Behavioral Research*, *51*, 519–539. PMID: 27314566. doi:10.1080/00273171.2016.1168279. eprint: http://dx.doi.org/10.1080/00273171.2016.1168279

O'Hara, R. B. & Sillanpää, M. J. (2009). A review of Bayesian variable selection methods: what, how and which. *Bayesian Analysis*, *4*, 85–117. doi:10.1214/09-BA403

Press, S. J. (2002). Bayesian factor analysis. In *Subjective and objective bayesian statistics* (pp. 359–390). John Wiley & Sons, Inc. doi:10.1002/9780470317105.ch15

Spiegelhalter, D. J., Best, N. G., Carlin, B. P., & van der Linde, A. (2014). The deviance information criterion: 12 years on. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *76*, 485–493. doi:10.1111/rssb.12062