Compensatory and Noncompensatory Multidimensional Randomized Item Response Analysis of the CAPS and EAQ Data

Fox, Klein Entink, Avetisyan BJMSP (March, 2013).

This document provides a step by step analysis of the CAPS and AEQ data using the *Compensatory and Noncompensatory Multidimensional Randomized Item Response Models*. The summarized output and description of the model is in Fox et al (2013, BJMSP). Here, a more detailed description is given of the function calls and the output. This document is not meant to serve as program manual, since it only provides information about the specific model analysis as described in the paper.

REAL Data Study: The R-work directory with stored data objects is called "DataObjects.Rdata"

```
load("DataObjects.Rdata") #load data set
```

The response matrix Y is defined to be N (793) times K (17) where the response options are integers from 1-5. The data object X contains a column with all ones (intercept), an indicator variable that equals zero when subject responded using the randomizing device (RRT = 0) and one when responded directly. The third column of X is an indicator that equals one when subject is female and zero when male. When there are no explanatory variables, an intercept still needs to be defined through a column of ones.

An RR object is made that specifies which response is answered via the randomized response procedure (RR=1) and which one via direct questioning (RR=0).

```
RR <- matrix(0,ncol=K, nrow=N)
RR[which(X[,2]==0),1:K] <- 1
```

The program is loaded via

```
## load MIRT-RR script
source("MIRT-RR.REAL.R")
```

The two-factor (Q=2) model is estimated through the following call making XG=50,000 iterations:

```
outq2      <- MIRTRR(Y,X,Q=2,XG=50000,RR)
```

Object outq2 will contain the sampled values from the posterior distributions. We can summarize results by computing means and variances.

To estimate the factor loadings, we use a burn-in of 5,000 iterations:

```
aa <- matrix(apply(outq2$MA[5000:50000,],2,mean), ncol=2, nrow=K)
```

The factor loadings can be standardized as follows

```
aa <- aa/sqrt(apply(aa**2,1,sum))
```

The factor loadings are defined to be positive for their main dimension, depending on the solution found. (When rescaling multiply theta with -1 and regression effects with -1).

```
aanames <- c("Item 1 ","Item 2 ","Item 3 ","Item 4 ","Item 5 ","Item 6 ","Item 7 ","Item 8 ","Item 9","Item 10 ","Item 11 ","Item 12 ","Item 13 ","Item 14 ","Item 15 ","Item 16 ","Item 17 ")

names(aa) <- NULL
rownames(aa) <- aanames

plot(aa[,2]/mdisc2,aa[,1]/mdisc2,xlab="Sexual Enhancement Expectancy ",ylab="Socio-Emotional/Community Problems", main="Estimated Standardized Factor Loadings",xlim=c(-0.25,1.1), ylim=c(.25,1.1)), text(aa[,2]/mdisc2, aa[,1]/mdisc2, row.names(aa), cex=0.8, pos=4, col="black")

abline(a=0,b=0,lwd=.75,lty=3)
```

This plot corresponds with ***Figure 2,*** which displays the estimated factor loadings.

Other estimates can be retrieved as follows and are given in *Table 3 under column "Two- Factor":*

```
# Covariance matrix Person parameters
matrix(apply((outq2$MSigmaP[10000:50000,]),2,mean),ncol=2) # posterior mean
     [,1]      [,2]
[1,]  0.98  0.65
[2,]  0.65  0.98
sqrt(matrix(apply((outq2$MSigmaP[10000:50000,]),2,var),ncol=2)) #posterior standard deviation

sqrt(matrix(apply((outq2n$MSigmaP[10000:50000,]),2,var),ncol=2))
        [,1]      [,2]
[1,] 0.05 0.07
[2,] 0.07 0.05

#Covariate Effects Factor One
apply(outq2$MBReg[10000:50000,,1],2,mean)  # posterior mean effects
0.00 (Intercept) -0.20 (Direct questioning) -0.00 (Female)
sqrt(apply(outq2$MBReg[10000:50000,,1],2,var))  # posterior variances
0.00 (Intercept) 0.09 (Direct-questioning) 0.06 (Female)

#Covariate Effects Factor Two
apply(outq2$MBReg[10000:50000,,2],2,mean)  # posterior mean effects
0.00 (Intercept)  -.22 (Direct questioning)  .03 (Female)
sqrt(apply(outq2$MBReg[10000:50000,,2],2,var))  # posterior variances
0.00 (Intercept) .06 (Direct questioning) .04 (Female)

#-2log-likelihood
mean(outq2$Mloglik[10000:50000])*(-2)

#Item thresholds
matrix(apply(outq2$Mbeta[10000:50000,],2,mean),nrow=K,ncol=4)
```

The three-factor (Q=3) model is estimated through the following call making XG=50,000 iterations:

```
outq3n    <- MIRTRR(Y,X,Q=3,XG=50000,RR)

aa <- matrix(apply(outq3n$MA[10000:50000,],2,mean),ncol=3,nrow=K)
aa <- aa/sqrt(apply(aa**2,1,sum))
```

These factor loadings are given in *Table 2*, where they are scaled to be positive for the main dimension they relate to.

Without giving again all results, the parameter estimates in *Table 3 under the label Three Factor* follow from the following commands:

```
## Covariance matrix Person parameters
matrix(apply((outq3n$MSigmaP[10000:50000,]),2,mean),ncol=3) # posterior means
sqrt(matrix(apply((outq3n$MSigmaP[10000:50000,]),2,var),ncol=3)) #posterior variances

#Covariate Effects and Variances Related to Factor One
apply(outq3n$MBReg[10000:50000,,1],2,mean)
sqrt(apply(outq3n$MBReg[10000:50000,,1],2,var))

#Covariate Effects and Variances Related to Factor Two
 apply(outq3n$MBReg[10000:50000,,2],2,mean)
 sqrt(apply(outq3n$MBReg[10000:50000,,2],2,var))

#Covariate Effects and Variances Related to Factor Three
apply(outq3n$MBReg[10000:50000,,3],2,mean)
sqrt(apply(outq3n$MBReg[10000:50000,,3],2,var))

#-2log-likelihood
mean(outq3n$Mloglik[10000:50000])*(-2)

## Define variable using effects-coding for clustering of students in universities

X3 <- matrix(0,ncol=3,nrow=N)
X3[which(data$SCHOOL==1),1] <- 1      #Elon
X3[which(data$SCHOOL==2),2] <- 1      #GTCC
X3[which(data$SCHOOL==3),3] <- 1      #Wake Forest
X3[which(data$SCHOOL==4),1:3] <- -1  #UNCG
X31 <- matrix(c(X[,-3],X3),ncol=5,nrow=N)
```

To obtain the results in *Table 4.*

```
outq2sch <- MIRTRR(Y,X31,Q=2,XG=50000,RR) #Two-factor model
```

The results are summarized in the same way as above.

```
outq3sch  <- MIRTRR(Y,X31,Q=3,XG=50000,RR) #Three-factor model
```

The results are summarized in the same way as above.

Explanation R-Code and additional results. Compensatory and non-compensatory Multidimensional Randomized Item Response Models.

This following describes the steps to arrive at the results presented in the section "Simulation Study" in the paper Fox, Klein Entink and Avetisyan (BJMSP, 2013). Furthermore, it provides some additional results that were not presented in the paper.

Step 1. Loading the R scripts

The following R-scripts are required:

1. Simdata.R. Contains the function to simulate data corresponding with the multidimensional randomized response model. Source this file in R via *source('simdata.R')*.
2. MIRT-RR.Simu.R Contains an adapted version to estimate the presented model that complies with the identification restrictions used to simulate the data. Source this file in R via *source('MIRT-RR.Simu.R')*.
3. Simulation study.R This file contains the specific R calls (and instructive comments) to run generate data, run the model and obtain the results.

Step 2. Simulate data

First, the number of desired items, persons, dimensions, response categories and covariates have to be initialised. In R, run:

```
N <- 750 ## number of test takers
K <- 20 ## number of items
C <- 4 ## number of response categories
Q <- 2 ## number of latent dimensions
P <- 3 ## number of covariates (including 1s for the intercept)
```

Then generate data with a call to the simdata() function as follows:

```
# generate data
sim <- simdata(N,K,C,Q=2,P)

## check the simulated data
sim$A # presents the loadings
sim$B # presents the simulated regression coefficients
sim$SigmaP # simulated covariance matrix person parameters.
```

Step3. Running the model

The model was ran twice to be able to obtain certain MCMC convergence diagnostics with a call to the MIRTRR() function. The output was stored in the list objects out1 and out2. In R run:

```
## run the model twice

out1 <- MIRTRR(Y=sim$Y,X= sim$X,Q=2,XG=10000)
out2 <- MIRTRR(Y=sim$Y,X= sim$X,Q=2,XG=10000)
```

Step 4. Analyzing the output

All the steps below can also be found in the simulation study.R script. The following commands were used to obtain the results presented in the tables:

The estimates for the covariance parameters:

```
## EAP estimates of the residual covariance matrices

colMeans(out1$MSigmaP)


## obtain SDs of covariance parameters

apply(out1$MSigmaP,2,sd)


## You can make some traceplots of specifc covariance parameters as follows:

plot(out1$MSigmaP[,1])
plot(out1$MSigmaP[,4])


## Obtain covariate regression parameter estimates and compare with simulated values

colMeans(out1$MBReg) # EAPs
apply(out1$MBReg,2,sd) # SDs
sim$B ## simulated values

## Making the traceplots as shown in the paper, with horizontal line depicting the true, simulated value.
```

Run the code below as follows and the figures will appear (also presented in Figure 1 below.)

```
trueB <- c(sim$B)[c(2,3,5,6)] ## the simulated values
BB <- out1$MBReg[5001:10000,,] ## obtain the last 5000 samples from the MCMC chain.
BB1 <- cbind(BB[,2:3,1],BB[,2:3,2]) ## restructure data matrix for ease of plotting.

#BB1 <- cbind(matrix(BB,ncol=1),sort(rep(1:4,1000)))
BB <- out2$MBReg[5001:10000,,]
BB2 <- cbind(BB[,2:3,1],BB[,2:3,2])
#BB2 <- cbind(matrix(BB,ncol=1),sort(rep(1:4,1000)))



#cbind(rep(1:1000,4),BB)
#BB <- data.frame(BB)
#p <- ggplot(BB, aes(x=BB[,1],y=BB[,2]))



y.name <- expression(gamma[11])
par(mfrow=c(2,2)) ## 2-by-2 layout.
plot(BB1[,1], type ="l", col = "red", xlab = "Iteration", ylab = y.name)
lines(BB2[,1],col = " blue")
abline(trueB[1],0, cex = 3)

y.name <- expression(gamma[12])
plot(BB1[,2], type ="l", col = "red", xlab = "Iteration", ylab = y.name)
lines(BB2[,2],col = " blue")
abline(trueB[2],0)

y.name <- expression(gamma[21])
plot(BB1[,3], type ="l", col = "red", xlab = "Iteration", ylab = y.name)
lines(BB2[,3],col = " blue")
abline(trueB[3],0)

y.name <- expression(gamma[22])
plot(BB1[,4], type ="l", col = "red", xlab = "Iteration", ylab = y.name)
lines(BB2[,4],col = " blue")
abline(trueB[4],0)
```

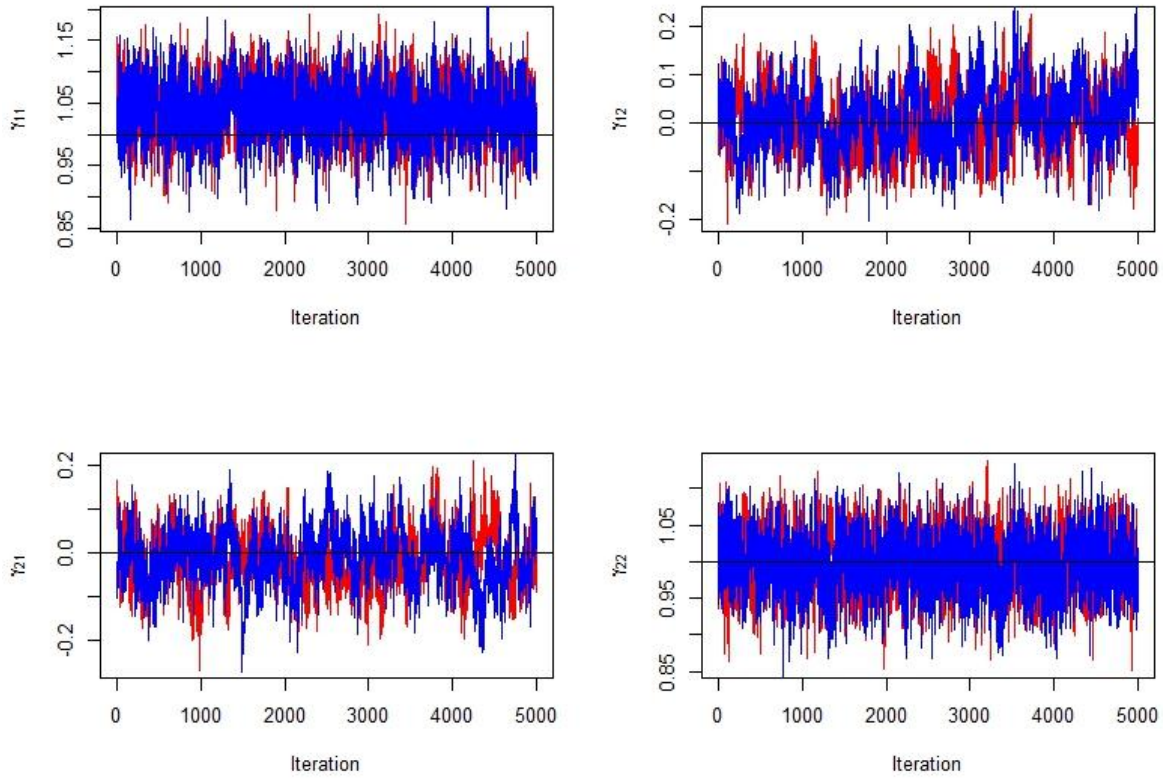

Figure 1: Traceplots for the regression coefficients

With the coda R package, convergence diagnostics can be obtained. We do this for the MCMC chains of the regression coefficients for illustration. The results are presented in Table 1 below.

Some convergence checks using the coda package.

```
library("coda")
BB1 <- as.mcmc(BB1)
BB2 <- as.mcmc(BB2)
gelman.diag(list(BB1,BB2))
```

Table 1: Gelman's diagnostics for the traceplots of the regression coefficients

```
                Point est. Upper C.I.
Gamma_11        1.01       1.03
Gamma_12        1.01       1.02
Gamma_21        1.01       1.04
Gamma_22        1.01       1.05


Multivariate psrf
1.02
```

To look at the estimated versus the simulated values of the person parameters, we generated Figure 2 below as follows:

```
### Make plots of simulated versus re-estimated ability parameters.

par(mfrow= c(1,2))
x.name <- expression(theta[1]*-simulated)
y.name <- expression(theta[1]*-EAP)
plot(sim$theta[,1],out1$EAPtheta[,1], xlab = x.name, ylab = y.name)
abline(a=0,b=1,col="red")
x.name <- expression(theta[2]*-simulated)
y.name <- expression(theta[2]*-EAP)
plot(sim$theta[,2],out1$EAPtheta[,2], xlab = x.name, ylab = y.name)
```

abline(a=0,b=1,col="red") abline(a=0,b=1,col="red")

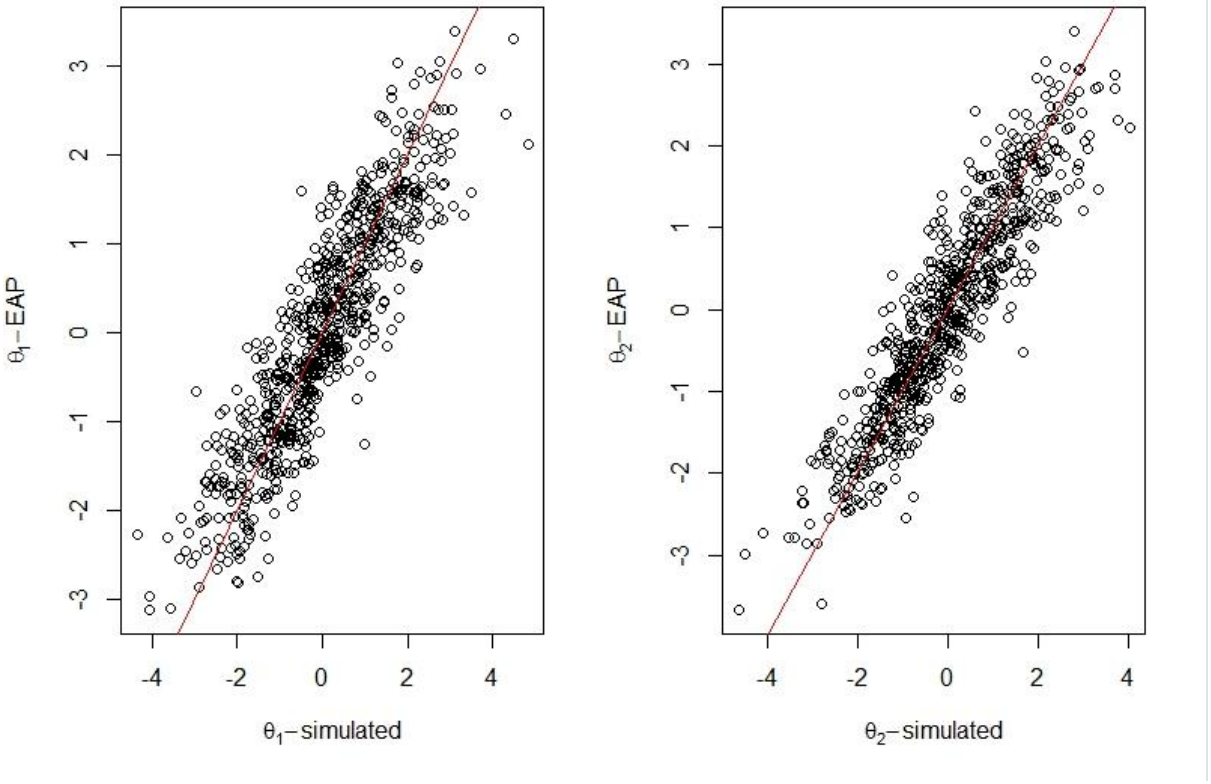

Figure 2: Simulated and re-estimated (EAP) ability parameters for the 750 persons. Red line is the identity line.

Similarly, a plot of the simulated versus the re-estimated factor loadings was made, as shown in Figure 3 below.

Plots of simulated versus re-estimated loadings.

plot(sim$A[2:20,1],colMeans(out1$MA)[2:20], xlab="simulated loadings dim 1", ylab="EAP loadings dim 1")
abline(0,1)

plot(sim$A[2:20,2],colMeans(out1$MA)[22:40], xlab="simulated loadings dim 2", ylab="EAP loadings dim 2")
abline(0,1)

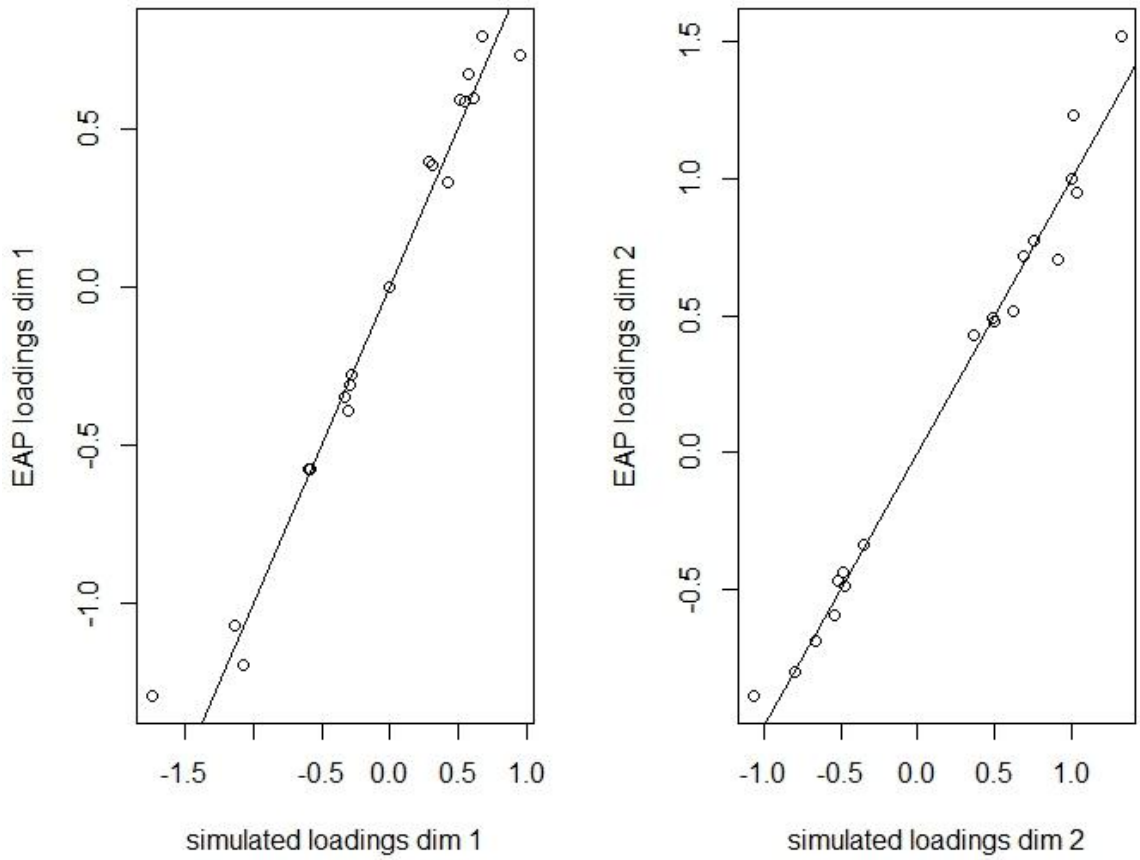

Figure 3: Simulated and re-estimated loadings for the two dimensions.