setwd("C:/R files BHMRA")

require(jagsUI)

require(HDInterval)

require(loo)

require(rstan);

attach("DS\_9\_6.Rdata")

set.seed=1234

#

# 2PL LATENT REGRESSION

#

# 0.432 is proportion of males in sample

cat("model {for (i in 1:n) { theta[i] ~ dnorm(delta\*(male[i]-0.432),1);

theta.new[i] ~ dnorm(delta\*(male[i]-0.432),1);

# item sampling model

for (j in 1:p){ y[i,j] ~ dbern(pi[i,j])

LL[j,i] <- y[i,j]\*log(pi[i,j])+(1-y[i,j])\*log(1-pi[i,j])

pi[i,j] <- ilogit(lambda[j]\*(theta[i]-alpha[j]))

# mixed replicates and predictive concordance

y.new[i,j] ~ dbern(pi.new[i,j])

pi.new[i,j] <- ilogit(lambda[j]\*(theta.new[i]-alpha[j]))

conc[i,j] <- equals(y[i,j],y.new[i,j])}}

# priors

for (j in 1:p) {lambda[j] ~ dnorm(0,tau.lambda) T(0,)

alpha[j] ~ dnorm(0,tau.alpha);

conc.item[j] <- sum(conc[,j])/n }

tau.alpha ~ dexp(1)

tau.lambda ~ dexp(1)

delta ~ dnorm(0,1)

# total information

for (t in 1:601) {for (j in 1:p) {pi.graph[j,t] <- ilogit(lambda[j]\*(theta.graph[t]-alpha[j]))

IF[j,t] <- lambda[j]^2\*pi.graph[j,t]\*(1-pi.graph[j,t]) }

TIF[t] <- sum(IF[,t])}}

", file="model1.jag")

**# Estimation**

inits1 <- list(lambda=rep(0.7,4),alpha=rep(0,4),theta=rep(0,658),delta=0)

inits2 <- list(lambda=rep(0.8,4), alpha=rep(0,4),theta=rep(0.2,658),delta=0.5)

inits=list(inits1,inits2)

pars=c("alpha","lambda","conc.item","delta","TIF")

R = autojags(DS\_9\_6, inits, pars,model.file="model1.jag",2,iter.increment=5000, n.burnin=500,Rhat.limit=1.1, max.iter=10000, seed=1234, codaOnly= c('TIF'))

R$summary

**# Information Function**

tifmn = apply(R$sims.list$TIF,2,mean)

tif10 = apply(R$sims.list$TIF,2,quantile,0.1)

tif90 = apply(R$sims.list$TIF,2,quantile,0.9)

plot(DS\_9\_6$theta.graph,tif90, xlab="Ability",ylab="Information",main="Figure 9.2 Test Information Function, Mean and 80% CRI",lwd=0.5)

lines(DS\_9\_6$theta.graph,tifmn, type = "o", col = "grey20")

lines(DS\_9\_6$theta.graph,tif10, type = "p", col = "grey60")

**# Fit Measures**

pars=c("LL")

R = autojags(DS\_9\_6, inits, pars,model.file="model1.jag",2,iter.increment=2500, n.burnin=7500,Rhat.limit=1.1, max.iter=10000, seed=1234, codaOnly= c('LL'))

sampsLL = as.array(R$sims.list$LL)

sampsLL=matrix(sampsLL,5000,4\*658)

loo(sampsLL)

waic(sampsLL)

**# 2PL Differential Item Functioning**

cat("model {for (i in 1:n) { theta[i] ~ dnorm(0,1);

theta.new[i] ~ dnorm(0,1);

# item sampling model

for (j in 1:p){ y[i,j] ~ dbern(pi[i,j])

LL[j,i] <- y[i,j]\*log(pi[i,j])+(1-y[i,j])\*log(1-pi[i,j])

pi[i,j] <- ilogit(lambda[G[i],j]\*(theta[i]-alpha[G[i],j]))

# mixed replicates and predictive concordance

y.new[i,j] ~ dbern(pi.new[i,j])

pi.new[i,j] <- ilogit(lambda[G[i],j]\*(theta.new[i]-alpha[G[i],j]))

conc[i,j] <- equals(y[i,j],y.new[i,j])}}

# priors

for (g in 1:2) {for (j in 1:p) {lambda[g,j] ~ dnorm(0,tau.lambda) T(0,)

alpha[g,j] ~ dnorm(0,tau.alpha)}}

for (j in 1:p) {conc.item[j] <- sum(conc[,j])/n }

tau.alpha ~ dexp(1)

tau.lambda ~ dexp(1)}

", file="model2.jag")

# Estimation

inits1 <- list(lambda= matrix(1,2,4),alpha= matrix(0,2,4),theta=rep(0,658))

inits2 <- list(lambda= matrix(0.5,2,4),alpha= matrix(-0.5,2,4),theta=rep(0.2,658))

inits=list(inits1,inits2)

pars=c("alpha","lambda","conc.item")

# Binary Gender as 1,2

DS\_9\_6$G=DS\_9\_6$male+1

R <- autojags(DS\_9\_6, inits, pars,model.file="model2.jag",2,iter.increment=5000, n.burnin=500,Rhat.limit=1.1, max.iter=50000, seed=1234)

R$summary

hdi(R,0.9)

**# Fit Measures**

pars=c("LL")

R = autojags(DS\_9\_6, inits, pars,model.file="model2.jag",2,iter.increment=2500, n.burnin=7500,Rhat.limit=1.1, max.iter=10000, seed=1234, codaOnly= c('LL'))

sampsLL = as.array(R$sims.list$LL)

sampsLL=matrix(sampsLL,5000,4\*658)

loo(sampsLL)

waic(sampsLL)

**# 2PL Differential Item Functioning, Reduced Model**

cat("model {for (i in 1:n) { theta[i] ~ dnorm(0,1);

theta.new[i] ~ dnorm(0,1);

# item sampling model

for (j in 1:p){ y[i,j] ~ dbern(pi[i,j])

LL[j,i] <- y[i,j]\*log(pi[i,j])+(1-y[i,j])\*log(1-pi[i,j])

pi[i,j] <- ilogit(lambda\*(theta[i]-alpha[G[i],j]))

# mixed replicates and predictive concordance

y.new[i,j] ~ dbern(pi.new[i,j])

pi.new[i,j] <- ilogit(lambda\*(theta.new[i]-alpha[G[i],j]))

conc[i,j] <- equals(y[i,j],y.new[i,j])}}

# priors

lambda ~ dnorm(0,1) T(0,)

tau.alpha ~ dexp(1)

rho ~ dunif(0.001,1000)

for (g in 1:2) {for (j in 1:p) {

alpha[g,j] ~ dnorm(0, tau.alpha /eta[g,j]^2)

# options on shrinkage parameters

# horseshoe

eta[g,j]~ dt(0, 1, 1) T(0,)

kap[g,j] <- 1/(1+ eta[g,j]^2)

# Lasso

# eta[g,j] <- sqrt(eta2[g,j])

# eta2[g,j] ~ dexp(rho^2/2)

}}

for (j in 1:p) {conc.item[j] <- sum(conc[,j])/n}}

", file="model3.jag")

# Estimation

inits1 <- list(lambda= 1,alpha= matrix(0,2,4),theta=rep(0,658))

inits2 <- list(lambda= 0.5,alpha= matrix(-0.5,2,4),theta=rep(0.2,658))

inits=list(inits1,inits2)

pars=c("alpha","lambda","conc.item","rho")

DS\_9\_6$G=DS\_9\_6$male+1

R <- autojags(DS\_9\_6, inits, pars,model.file="model3.jag",2,iter.increment=5000, n.burnin=500,Rhat.limit=1.1, max.iter=50000, seed=1234)

R$summary

hdi(R,0.9)

# LOO-IC

pars=c("LL")

R = autojags(DS\_9\_6, inits, pars,model.file="model3.jag",2,iter.increment=2500, n.burnin=7500,Rhat.limit=1.1, max.iter=10000, seed=1234, codaOnly= c('LL'))

sampsLL = as.array(R$sims.list$LL)

sampsLL=matrix(sampsLL,5000,4\*658)

loo(sampsLL)

waic(sampsLL)

**#**

**# Multivariate Probit**

**#**

# K = number predictors (excluding constant), P = number of indicators

x=cbind(1,DS\_9\_6$male)

ytr <- matrix(1,658,4)

y <- DS\_9\_6$y

# change (0,1) coding to (-1,1) coding

for (j in 1:4) {ytr[y[,j]==0,j] <- -1}

Dstan=list(y=ytr,x=x,P=4,K=2,n=658)

MVP.stan <- "

data { int<lower=1> K;

int<lower=1> P;

int<lower=1> n;

matrix<lower=-1,upper=1>[n,P] y;

matrix[n,K] x;

}

parameters { matrix[K,P] beta;

cholesky\_factor\_corr[P] L\_R;

matrix<lower=0>[n,P] abs\_ystar; //continuous latent variable

}

transformed parameters { matrix[n,P] z;

z = (y .\* abs\_ystar);

}

model { L\_R ~ lkj\_corr\_cholesky(1);

to\_vector(beta) ~ normal(0,1);

{

matrix[n,P] beta\_x;

beta\_x = x \* beta;

for (i in 1:n)

z[i,] ~ multi\_normal\_cholesky(beta\_x[i,], L\_R);} }

generated quantities { matrix[P,n] z\_new;

matrix[P,P] R;

R = multiply\_lower\_tri\_self\_transpose(L\_R);

{ matrix[P,n] beta\_x;

beta\_x = (x \* beta)';

for (i in 1:n)

z\_new[,i] = multi\_normal\_cholesky\_rng(beta\_x[,i], L\_R); } }

"

sm <- stan\_model(model\_code= MVP.stan)

fit <- sampling(sm, data =Dstan, iter = 2000,warmup=200,chains = 2)

summary(fit, pars = c("beta","R"), probs = c(0.025,0.05, 0.95, 0.975))$summary