set.seed=1234

setwd("C:/R files BHMRA")

require(rstan)

require(boot)

require(loo)

attach("DS\_9\_5.Rdata")

**# Two Parameter Logit (2PL)**

set.seed=1234

IRT2PL.stan <- "

data { int n;

int p;

int<lower=0,upper=1> y[n,p];

}

parameters { vector[n] theta;

vector<lower=0> [p] lambda;

vector[p] alpha;

real<lower=0>sigma\_lambda;

real<lower=0>sigma\_alpha;

}

model {theta ~ normal(0,1);

alpha ~ normal(0,sigma\_alpha);

sigma\_alpha ~ cauchy(0,5);

lambda ~ lognormal(0,sigma\_lambda);

sigma\_lambda ~ cauchy(0,5);

for (j in 1:p){

y[,j] ~ bernoulli\_logit(lambda[j]\*(theta - alpha[j])) ;}

}

generated quantities {vector[p] loglik[n];

for (i in 1: n){for (j in 1: p){

real pi;

pi= inv\_logit(lambda[j]\*(theta[i] -alpha[j]));

loglik[i, j] = bernoulli\_log(y[i, j],pi);

}}}

"

**# ESTIMATION**

sm2 <- stan\_model(model\_code=IRT2PL.stan)

post2 <- sampling(sm2, data =DS\_9\_5, iter = 2000,warmup=500,chains = 2,seed=1234)

**# Parameters**

summary(post2, pars = c("lambda","alpha"), probs = c(0.025,0.05, 0.95, 0.975))$summary

**# Fit Measure**

LOO2=loo(as.matrix(post2,pars="loglik"))

WAIC2=waic(as.matrix(post2,pars="loglik"))

**# TEST INFORMATION FUNCTION PLOT**

theta.graph = seq(-4, 2, length.out = 601)

lambda.samp = as.matrix(post2,pars="lambda")

alpha.samp = as.matrix(post2,pars=" alpha")

IF = array(,dim=c(1500,5,601))

IF.t = matrix(,1500,601)

**# information function by sample s, ability theta.graph[t], item j**

for (s in 1:1500) {for(t in 1:601) {for (j in 1:5){

pi = inv.logit(lambda.samp[s,j]\*(theta.graph[t]-alpha.samp[s,j]))

IF[s,j,t] = lambda.samp[s,j]^2\*pi\*(1-pi) }

# total IF

IF.t[s,t] = sum(IF[s,,t])}}

**# Item Specific Functions**

IF.item = matrix(,5,601)

for(t in 1:601){for (j in 1:5){IF.item[j,t] <- mean(IF[,j,t])}}

**# Total Information Function**

IF.t.m=apply(IF.t,2,mean)

IF.t.q20=apply(IF.t,2,quantile,0.2)

IF.t.q80=apply(IF.t,2,quantile,0.8)

plot(theta.graph,IF.t.q80,xlab="Ability",ylab="Test Information",main="Figure 9.1 Test Information Plot (Mean and 60% CRI)", col = "grey80")

lines(theta.graph,IF.t.m, type = "o", col = "grey20")

lines(theta.graph,IF.t.q20, type = "o", col = "grey60")

**# Predictive Concordance**

y <- DS\_9\_5$y

thetas <- as.matrix(post2,pars="theta")

ynew <- array(,dim=c(1000,5,1500))

match <- array(,dim=c(1000,5,1500))

predmatch <- numeric(5)

**# iterations (s), items (j), subjects (i)**

for (s in 1:1500){for (j in 1:5) {for(i in 1:1000) {

pi <- inv.logit(lambda.samp[s,j]\*(thetas[s,i]-alpha.samp[s,j]))

**# sample predictions**

ynew[i,j,s] = rbinom(1, 1, pi)

match[i,j,s] = ifelse(ynew[i,j,s]==y[i],1,0)}}}

for (j in 1:5) {predmatch[j] =mean(match[,j,])}

**# 3PL IRT, Hierarchical Prior on Inverse Logit of Threshold Parameters**

set.seed=1234

IRT3PL.stan <- "

data { int n;

int p;

int<lower=0,upper=1> y[n,p];

}

parameters { vector[n] theta;

vector<lower=0> [p] lambda;

vector[p] alpha;

vector[p] xi;

real<lower=0>sigma\_lambda;

real<lower=0>sigma\_alpha;

real<lower=0>sigma\_xi;

}

transformed parameters {

vector[p] gamma;

gamma= inv\_logit(xi);

}

model {theta ~ normal(0,1);

alpha ~ normal(0,sigma\_alpha);

sigma\_alpha ~ cauchy(0,5);

lambda ~ lognormal(0,sigma\_lambda);

sigma\_lambda ~ cauchy(0,5);

xi ~ normal(0,sigma\_xi);

sigma\_xi ~ cauchy(0,5);

for(i in 1:n){for (j in 1:p){ real pi; //pi is local variable

pi= inv\_logit(lambda[j]\*(theta[i] - alpha[j]));

y[i,j] ~ bernoulli(gamma[j] + (1-gamma[j])\*pi);}}

}

generated quantities {vector[p] loglik[n];

for (i in 1: n){for (j in 1: p){real pi;

pi= inv\_logit(lambda[j]\*(theta[i] -alpha[j]));

loglik[i, j] = bernoulli\_log(y[i, j], gamma[j] + (1-gamma[j])\*pi);}}}

"

**# ESTIMATION**

sm3 <- stan\_model(model\_code=IRT3PL.stan)

post3 <- sampling(sm3, data =DS\_9\_5, iter = 5000,warmup=500,chains = 2,seed=1234)

**# Parameters**

summary(post3, pars = c("lambda","alpha","gamma"), probs = c(0.025,0.05, 0.95, 0.975))$summary

**# Fit measures**

LOO3=loo(as.matrix(post3,pars="loglik"))

WAIC3=waic(as.matrix(post3,pars="loglik"))

**# 3PL IRT Fixed Effects (Beta Prior) on Threshold Parameters**

set.seed=1234

IRT3PLBeta.stan <- "

data { int n;

int p;

int<lower=0,upper=1> y[n,p];

}

parameters { vector[n] theta;

vector<lower=0> [p] lambda;

vector[p] alpha;

vector<lower=0,upper=1> [p] gamma; //threshold parameters

real<lower=0>sigma\_lambda;

real<lower=0>sigma\_alpha;

real<lower=0>a1;

real<lower=0>b1;

}

model {theta ~ normal(0,1);

alpha ~ normal(0,sigma\_alpha);

sigma\_alpha ~ cauchy(0,5);

lambda ~ lognormal(0,sigma\_lambda);

sigma\_lambda ~ cauchy(0,5);

gamma ~ beta(a1,b1);

a1 ~ exponential(1);

b1 ~ exponential(1);

for(i in 1:n){for (j in 1:p){ real pi; //pi is local variable

pi= inv\_logit(lambda[j]\*(theta[i] - alpha[j]));

y[i,j] ~ bernoulli(gamma[j] + (1-gamma[j])\*pi);}}

}

generated quantities {vector[p] loglik[n];

for (i in 1: n){for (j in 1: p){real pi;

pi= inv\_logit(lambda[j]\*(theta[i] -alpha[j]));

loglik[i, j] = bernoulli\_log(y[i, j], gamma[j] + (1-gamma[j])\*pi);}}}

"

# Estimation

sm4 <- stan\_model(model\_code=IRT3PLBeta.stan)

post4 <- sampling(sm4, data =DS\_9\_5, iter = 5000,warmup=500,chains = 2,seed=1234)

**# Parameters**

summary(post4, pars = c("lambda","alpha","gamma"), probs = c(0.025,0.05, 0.95, 0.975))$summary

**# Fit measures**

LOO4=loo(as.matrix(post4,pars="loglik"))

WAIC4=waic(as.matrix(post4,pars="loglik"))