options(scipen=999)

library(jagsUI)

library(loo)

library(rstan)

library(bridgesampling)

D=list(y=c(10,23,23,26,17,5,53,55,32,46,10,8,10,8,23,0,3,22,15,32,3),

n=c(39,62,81,51,39,6,74,72,51,79,13,16,30,28,45,4,12,41,30,51,7),

x1=c(0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1),

x2=c(0,0,0,0,0,1,1,1,1,1,1,0,0,0,0,0,1,1,1,1,1),N=21)

**# Model 1, No RE Selection**

cat("

model {for( i in 1 : N ) { y[i] ~ dbin(p[i],n[i])

b[i] ~ dnorm(0,tau.b)

ind.b[i] <- step(b[i])

LL[i] <- logfact(n[i])-logfact(y[i])-logfact(n[i]-y[i])+y[i]\*log(p[i])

+(n[i]-y[i])\*log(1-p[i])

logit(p[i]) <- beta[1]+beta[2]\*x1[i]+beta[3]\*x2[i]

+beta[4]\*x1[i]\*x2[i]+b[i]}

for (j in 1:4) {beta[j] ~ dnorm(0,0.001)}

tau.b ~ dgamma(1,0.001)

sigma.b <- 1 / sqrt(tau.b)}

", file="model1.jag")

**# Estimation**

inits1 = list(tau.b=10, beta=rep(0,4))

inits2 = list(tau.b=100, beta=rep(0,4))

inits=list(inits1,inits2)

pars = c("beta","b","ind.b","sigma.b","LL")

R1 = autojags(D, inits, pars,model.file="model1.jag",2,iter.increment=100, n.burnin=100,Rhat.limit=1.1, max.iter=2000, seed=1234, codaOnly= c('LL'))

R1$summary

# Fit

loo(as.matrix(R1$sims.list$LL))

**# Model 2, RE Selection, Laplace Mixture**

D$r=0.00001

cat(" model {for( i in 1 : N ) { y[i] ~ dbin(p[i],n[i])

LL[i] <- logfact(n[i])-logfact(y[i])-logfact(n[i]-y[i])+y[i]\*log(p[i])+(n[i]-y[i])\*log(1-p[i])

logit(p[i])<-beta[1]+beta[2]\*x1[i]+beta[3]\*x2[i]+beta[4]\*x1[i]\*x2[i]+b[i]}

for (j in 1:4) {beta[j] ~ dnorm(0,0.001)}

# selection of random effects

for (i in 1:N) {b[i] ~ dnorm(0,1/psi[i])

ind.b[i] <- step(b[i])

delta[i] ~ dbern(omega)

# variance of random effects under spike-slab

psi[i] <- equals(delta[i],0)\*psi.spk+equals(delta[i],1)\*psi.slb}

# total random effects retained

sum.delta <- sum(delta[])

# hyperparameters

invQ ~ dgamma(0.5,0.2275)

omega ~ dbeta(1,1)

Q <- 1/invQ

psi.slb ~ dexp(0.5/Q)

psi.spk ~ dexp(0.5/(Q\*r))}

", file="model2.jag")

**# Estimation**

inits1 = list(invQ=10, beta=rep(0,4))

inits2 = list(invQ=100, beta=rep(0,4))

inits=list(inits1,inits2)

pars = c("beta","b","delta","ind.b","LL","sum.delta")

R2 = autojags(D, inits, pars,model.file="model2.jag",2,iter.increment=2500, n.burnin=100,Rhat.limit=1.1, max.iter=50000, seed=1234,codaOnly= c('LL'))

R2$summary

**# Fit**

loo(as.matrix(R2$sims.list$LL))

**# Retention Probabilities**

delta.samps= as.matrix(R2$sims.list$delta)

range(apply(delta.samps,2,mean))

**# Probabilities of high random effects**

ind.b.samps =as.matrix(R2$sims.list$ind.b)

range(apply(ind.b.samps,2,mean))

hist(apply(ind.b.samps,2,mean),xlab="Probability",main="Figure 3.3 Probabilties of High Random Effects",col="gray")

**#**

**# rstan estimation, Model 1, No RE Selection**

**#**

M1\_code = "data {

int<lower=0> N;

int<lower=0> y[N];

int<lower=0> n[N];

vector[N] x1;

vector[N] x2;

}

transformed data {

vector[N] x1x2;

x1x2 = x1 .\* x2;

}

parameters {

real beta[4];

real<lower=0> tau\_b;

vector[N] b;

}

transformed parameters {

real<lower=0> sigma\_b;

sigma\_b = 1.0 / sqrt(tau\_b);

}

model {

beta ~ normal(0,5);

tau\_b ~ gamma(1,1.0E-3);

b ~ normal(0.0, sigma\_b);

y ~ binomial\_logit(n, beta[1] + beta[2] \* x1 + beta[3] \* x2 + beta[4] \* x1x2 + b);

}"

**# Estimation**

set.seed(1)

fit\_M1 <- stan(model\_code = M1\_code,data=D, iter = 5000, warmup = 500, chains = 2, seed = 1)

**# Model 3, Horseshoe Prior on Random Intercepts**

M3\_code = "data {

int<lower=0> N;

int<lower=0> y[N];

int<lower=0> n[N];

vector[N] x1;

vector[N] x2;

}

transformed data {

vector[N] x1x2;

x1x2 = x1 .\* x2;

}

parameters {

real beta[4];

vector[N] b;

vector<lower=0>[N] lambda;

real<lower=0> tau2;

}

transformed parameters {

vector<lower=0>[N] kappa;

real <lower=0> tau;

tau = sqrt(tau2);

// effective retention rate

for (i in 1:N) {kappa[i] = lambda[i]^2/(1+lambda[i]^2);}

}

model {

lambda ~ cauchy(0,1);

tau2 ~ cauchy(0,1);

for (i in 1:N) {b[i] ~ normal(0, lambda[i] \* tau);}

beta ~ normal(0,5);

y ~ binomial\_logit(n, beta[1] + beta[2] \* x1 + beta[3] \* x2 + beta[4] \* x1x2 + b);

}"

**# Estimation**

set.seed(1)

fit\_M3 <- stan(model\_code = M3\_code,data=D, iter = 10000, warmup = 500, chains = 2, seed = 1)

print(fit\_M3,digits=3)

kappa.samps = as.matrix(fit\_M3, pars = "kappa")

range(apply(kappa.samps,2,mean))

hist(apply(kappa.samps,2,mean),xlab="Weight",main="Figure 3.4 Histogram of Weights for Non-Zero Effects, Horseshoe Prior",cex.main=0.9,col="gray")

**# Formal Comparison, M1 and M3**

bridge\_M1 <- bridge\_sampler(fit\_M1)

bridge\_M3 <- bridge\_sampler(fit\_M3)

error\_measures(bridge\_M1)$percentage

error\_measures(bridge\_M3)$percentage

bf(bridge\_M1, bridge\_M3)

bridge\_M1

bridge\_M3