require(jagsUI)

require(loo)

library(bridgesampling)

library(rjags)

library(rstan)

data("turtles")

D= list(n=244,p=2,x=turtles$x,y=turtles$y)

**# MODEL 1 REGRESSION COEFFICIENTS ONLY**

cat("model {for (i in 1:n) {

y[i] ~ dbern(pi[i]);

pi[i] <- phi(nu[i])

nu[i] <- beta[1] + beta[2]\*x[i];

**# log-likelihood**

LL[i] <- y[i]\*log(phi(nu[i])) + (1-y[i])\*log(1-phi(nu[i]))}

Dv <- -2\*sum(LL[])

**# priors**

for (j in 1:p) {beta[j] ~ dnorm(0,0.01)}}

", file="model1.jag")

inits1 = list(beta=rep(0.1,2))

inits2 = list(beta=rep(-0.1,2))

inits=list(inits1,inits2)

**# DIC with different penalties**

M1 = jags.model(inits=inits,data=D,n.chains=2, file="model1.jag")

update(M1,500)

DIC1.pD= dic.samples(M1, n.iter=10000, type="pD")

DIC1.popt= dic.samples(M1, n.iter=10000, type="popt")

cat("\n","DIC Measures, Model 1","\n")

DIC1.pD

cat("\n","DIC Measures, Model 1","\n")

DIC1.popt

**# WAIC and LOO-IC**

pars = c("LL","beta")

R1 = autojags(D, inits, pars,model.file="model1.jag",2,iter.increment=1000, n.burnin=100,Rhat.limit=1.025, max.iter=5000, seed=1234,codaOnly= c('LL'))

R1$summary

sampsLL <- as.matrix(R1$sims.list$LL)

LOO1=loo(as.matrix(sampsLL))

loo.pw =as.vector(LOO1$pointwise[,3])

WAIC1=waic(as.matrix(sampsLL))

waic.pw =as.vector(WAIC1$pointwise[,3])

**# WAIC by calculation**

sampsL=exp(sampsLL)

waic1=log(apply(sampsL,2,mean))

waic2=apply(sampsLL,2,sd)^2

waic.pw=-2\*(waic1-waic2)

elpd\_waic=sum(waic1)-sum(waic2)

waic=-2\*elpd\_waic

**# pointwise lack of fit, % WAIC due to 5% worst fitting cases**

100\*sum(waic.pw[waic.pw>quantile(waic.pw,0.95)])/waic

**# worst fitting cases by clutch**

clutch=turtles$clutch

list = data.frame(waic.pw,clutch)

list=list[order(-list$waic),]

head(list,20)

**# MODEL 2 RANDOM CLUTCH INTERCEPTS**

J=31

D$J=J

D$clutch=clutch

**# code**

cat("

model {for (i in 1:n) { y[i] ~ dbern(phi(nu[i]))

nu[i] <- beta[1] + beta[2]\*x[i] + b[clutch[i]]

# log-likelihoods

LL[i] <- y[i]\*log(phi(nu[i])) + (1-y[i])\*log(1-phi(nu[i]))}

for (j in 1:p) {beta[j] ~ dnorm(0,0.1)}

# gamma prior on precision

# tau.b ~ dgamma(0.1,0.1); sigma2.b <- 1/tau.b

# shrinkage prior on clutch variance

U ~ dunif(0,1)

sigma2.b <- (1-pow(U,0.5))/pow(U,0.5)

tau.b <- 1/sigma2.b

for (j in 1:J) { b[j] ~ dnorm(0,tau.b)

ind.b[j] <- step(b[j])}}

", file="model2.jag")

inits1 = list(b=rep(0,J),beta=rep(-0.1,2))

inits2 = list(b=rep(0,J),beta=rep(0.1,2))

inits=list(inits1,inits2)

**# DIC with different penalties**

M2 = jags.model(inits=inits,data=D,n.chains=2, file="model2.jag")

update(M2,500)

DIC2.pD= dic.samples(M2, n.iter=10000, type="pD")

DIC2.popt= dic.samples(M2, n.iter=10000, type="popt")

cat("\n","DIC Measures, Model 2","\n")

DIC2.pD

cat("\n","DIC Measures, Model 2","\n")

DIC2.popt

**# WAIC and LOO-IC, MODEL 2**

pars = c("LL","beta","sigma2.b","ind.b")

R2 = autojags(D, inits, pars,model.file="model2.jag",2,iter.increment=2500, n.burnin=100,Rhat.limit=1.025, max.iter=10000, seed=1234,codaOnly= c('LL'))

sampsLL <- as.matrix(R2$sims.list$LL)

LOO2=loo(as.matrix(sampsLL))

LOO2

loo.pw =as.vector(LOO2$pointwise[,3])

WAIC2=waic(as.matrix(sampsLL))

WAIC2

waic.pw =as.vector(WAIC2$pointwise[,3])

**# pointwise lack of fit, % WAIC due to 5% worst fitting cases**

100\*sum(waic.pw[waic.pw>quantile(waic.pw,0.95)])/waic

**# Density of Clutch Variance**

d=density(as.matrix(R2$sims.list$sigma2.b))

plot(d,main="Figure 3.1 Density of Clutch Variance",xlab="Variance")

polygon(d, col="gray", border="black")

**# MODEL 3 RANDOM CLUTCH SLOPES**

cat("

model {for (i in 1:n) { y[i] ~ dbern(phi(nu[i]))

nu[i] <- beta[1] + b[clutch[i]]\*x[i]

# log-likelihoods

LL[i] <- y[i]\*log(phi(nu[i])) + (1-y[i])\*log(1-phi(nu[i]))}

for (j in 1:p) {beta[j] ~ dnorm(0,0.1)}

# gamma prior on precision

# tau.b ~ dgamma(0.1,0.1); sigma2.b <- 1/tau.b

# shrinkage prior on clutch variance

U ~ dunif(0,1)

sigma2.b <- (1-pow(U,0.5))/pow(U,0.5)

tau.b <- 1/sigma2.b

for (j in 1:J) { b[j] ~ dnorm(beta[2],tau.b)

ind.b[j] <- step(b[j]-beta[2])}}

", file="model3.jag")

inits1 = list(b=rep(0,J),beta=rep(-0.1,2))

inits2 = list(b=rep(0,J),beta=rep(0.1,2))

inits=list(inits1,inits2)

**# DIC with different penalties**

M3 = jags.model(inits=inits,data=D,n.chains=2, file="model3.jag")

update(M3,500)

DIC3.pD= dic.samples(M3, n.iter=10000, type="pD")

DIC3.popt= dic.samples(M3, n.iter=10000, type="popt")

cat("\n","DIC Measures, Model 3","\n")

DIC3.pD

cat("\n","DIC Measures, Model 3","\n")

DIC3.popt

**# WAIC and LOO-IC**

pars = c("LL","beta","sigma2.b","ind.b","b")

R3 = autojags(D, inits, pars,model.file="model3.jag",2,iter.increment=2500, n.burnin=100,Rhat.limit=1.025, max.iter=10000, seed=1234,codaOnly= c('LL'))

sampsLL <- as.matrix(R3$sims.list$LL)

LOO3=loo(as.matrix(sampsLL))

LOO3

loo.pw =as.vector(LOO3$pointwise[,3])

WAIC3=waic(as.matrix(sampsLL))

WAIC3

waic.pw =as.vector(WAIC3$pointwise[,3])

**# Frequency plot, random slopes**

hist(R3$mean$b,xlab="Slope",main="Figure 3.2 Random Slopes on Birth Weight",breaks=15,col="gray")

#

**# rstan for WBIC, model 1**

#

M1="data {

int<lower = 1> N;

real logN;

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

real<lower = 0> x[N];

}

parameters {

real alpha;

real beta;

}

transformed parameters {

vector[N] LL;

for ( i in 1:N ) {LL[i] = bernoulli\_lpmf(y[i] | Phi(alpha+beta\*x[i])); }

}

model {

target += normal\_lpdf(alpha | 0, 3.16);

target += normal\_lpdf(beta | 0, 3.16);

for (i in 1:N) { target += bernoulli\_lpmf(y[i] | Phi(alpha+beta\*x[i]))/logN; }

}

generated quantities {

real log\_like;

log\_like = sum(LL);

}"

N=244

D1=list(y = turtles$y,x = turtles$x,N=N,logN=log(N))

fit=stan(model\_code=M1,data=D1,iter=1250,warmup=250,chains=1,seed=10)

WBIC=c()

WBIC[1]= -2\*summary(fit, pars = c("log\_like"))$summary[1]

#

**# rstan for WBIC, model 2**

#

M2="

data {

int<lower = 1> N;

real logN;

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

real<lower = 0> x[N];

int<lower = 1> C;

int<lower = 1, upper = C> clutch[N];

}

parameters {

real alpha;

real beta;

vector[C] b\_n01;

real<lower = 0> sigma2;

}

transformed parameters {

vector[C] b;

vector[N] LL;

real<lower = 0> sigma = sqrt(sigma2);

b = sigma \* b\_n01;

for ( i in 1:N ) {LL[i] = bernoulli\_lpmf(y[i] | Phi(alpha+beta\*x[i]+b[clutch[i]])); }

}

model {

target += - 2 \* log(1 + sigma2); // p(sigma2) = 1 / (1 + sigma2) ^ 2

target += normal\_lpdf(alpha | 0, 3.16); // prior

target += normal\_lpdf(beta | 0, 3.16); // prior

target += normal\_lpdf(b\_n01 | 0, 1); // random effects

for (i in 1:N) { target += bernoulli\_lpmf(y[i] | Phi(alpha+beta\*x[i]+b[clutch[i]]))/logN; }

}

generated quantities {

real log\_like;

log\_like = sum(LL);

}

"

N=244

D2=list(y = turtles$y,x = turtles$x,N =N,logN=log(N),C = max(turtles$clutch),clutch = turtles$clutch)

fit=stan(model\_code=M2,data=D2,iter=1250,warmup=250,chains=1,seed=10)

WBIC[2]= -2\*summary(fit, pars = c("log\_like"))$summary[1]

#

**# rstan for WBIC, model 3**

#

M3="

data {

int<lower = 1> N;

real logN;

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

real<lower = 0> x[N];

int<lower = 1> C;

int<lower = 1, upper = C> clutch[N];

}

parameters {

real alpha;

real beta;

vector[C] b\_n01;

real<lower = 0> sigma2;

}

transformed parameters {

vector[C] b;

vector[N] LL;

real<lower = 0> sigma = sqrt(sigma2);

b = sigma \* b\_n01;

for ( i in 1:N ) {LL[i] = bernoulli\_lpmf(y[i] | Phi(alpha+(beta+b[clutch[i]])\*x[i])); }

}

model {

target += - 2 \* log(1 + sigma2); // p(sigma2) = 1 / (1 + sigma2) ^ 2

target += normal\_lpdf(alpha | 0, 3.16); // prior

target += normal\_lpdf(beta | 0, 3.16); // prior

target += normal\_lpdf(b\_n01 | 0, 1); // random effects

for (i in 1:N) { target += bernoulli\_lpmf(y[i] | Phi(alpha+(beta+b[clutch[i]])\*x[i]))/logN; }

}

generated quantities {

real log\_like;

log\_like = sum(LL);

}

"

fit=stan(model\_code=M3,data=D2,iter=1250,warmup=250,chains=1,seed=10)

WBIC[3]= -2\*summary(fit, pars = c("log\_like"))$summary[1]