require(jagsUI)

library(bridgesampling)

library(rstan)

data("turtles")

options(scipen=999)

**#**

**# COMPARISONS USING JAGSUI**

**#**

# define w according to length, T, of bridge-sampling schedule

T=10

# number of observations

n=244

T1=T+1

D= list(T=T,T1=T1, w=matrix(1,n,T1),n=n,p=2, x=as.numeric(scale(turtles$x)),

y=turtles$y,path.pow=4,

a=c(0.00025,0.0005,0.001,0.005,0.05,0.075,0.1,0.25,0.5,0.75,0.99))

**# MODEL 1 FIXED EFFECTS (REGRESSION COEFFICIENTS) ONLY**

cat("

model {for (h in 1:n) {for (s in 1:T1) {

L.tem[h,s] <- pow(L[h,s],q[s])

w[h,s] ~ dunif(a1[h,s],b1[h,s])

a1[h,s] <- -1/L.tem[h,s]

b1[h,s] <- 1/L.tem[h,s]

LL[h,s] <- log(L[h,s])

pi[h,s] <- phi(beta1[s]+beta2[s]\*x[h])

# log-likelihood

log(L[h,s]) <- y[h]\*log(pi[h,s])+(1-y[h])\*log(1-pi[h,s])}}

# regression parameters

est.beta[1] <- beta1[T1]

est.beta[2] <- beta2[T1]

for (s in 1:T1) {beta1[s] ~ dnorm(0,1)

beta2[s] ~ dnorm(0,1)

# path sampling calculations

q[s] <- pow(a[s],path.pow)

expLL[s] <- sum(LL[1:n,s]) }

for (s in 1:T) {mc[s] <- (q[s+1]-q[s])\*(expLL[s+1]+expLL[s])\*0.5 }

logML <- sum(mc[])}

", file="model1.jag")

inits1 = list(beta1=rep(-0.5,T1), beta2=rep(0.4,T1))

inits2 = list(beta1=rep(-0.6,T1),beta2=rep(0.35,T1))

inits=list(inits1,inits2)

pars = c("logML","est.beta")

R1 = autojags(D, inits, pars,model.file="model1.jag",2,iter.increment=500, n.burnin=100,Rhat.limit=1.1, max.iter=2500, seed=1234)

R1$summary

**# MODEL 2 RANDOM CLUTCH INTERCEPTS**

**# Gamma Prior on Clutch Precision**

J=31

D$J=J

D$clutch=turtles$clutch

cat(" model {for (h in 1:n) {for (s in 1:T1) {

L.tem[h,s] <- pow(L[h,s],q[s])

w[h,s] ~ dunif(a1[h,s],b1[h,s])

a1[h,s] <- -1/L.tem[h,s];

b1[h,s] <- 1/L.tem[h,s]

LL[h,s] <- log(L[h,s])

pi[h,s] <- phi(beta1[s]+beta2[s]\*x[h]+b[clutch[h],s])

# log-likelihood

log(L[h,s]) <- y[h]\*log(pi[h,s])+(1-y[h])\*log(1-pi[h,s])}}

# priors over path (1,...,T+1)

# regression parameters

est.beta[1] <- beta1[T1]

est.beta[2] <- beta2[T1]

for (s in 1:T1) {beta1[s] ~ dnorm(0,1)

beta2[s] ~ dnorm(0,1)}

# random effects

for (j in 1:J) { est.b[j] <- b[j,T1]

for (s in 1:T1) {b[j,s] ~ dnorm(0,tau.b[s])}}

# random effect variance

est.s2 <- 1/tau.b[T1]

for (s in 1:T1) {tau.b[s] ~ dgamma(0.1,0.1)

# path sampling calculations

q[s] <- pow(a[s],path.pow)

expLL[s] <- sum(LL[1:n,s]) }

for (s in 1:T) {mc[s] <- (q[s+1]-q[s])\*(expLL[s+1]+expLL[s])\*0.5 }

logML <- sum(mc[])}

", file="model2.jag")

inits1 = list(b=matrix(0,J,T1), beta1=rep(-0.4,T1), beta2=rep(0.4,T1),tau.b=rep(10,T1))

inits2 = list(b=matrix(0,J,T1), beta1=rep(-0.5,T1), beta2=rep(0.35,T1),tau.b=rep(100,T1))

inits=list(inits1,inits2)

pars = c("logML","est.b","est.s2","est.beta")

R2 = autojags(D, inits, pars,model.file="model2.jag",2,iter.increment=500, n.burnin=100,Rhat.limit=1.1, max.iter=2500, seed=1234)

R2$summary

**# posterior mean random effects**

est.b=R2$summary[2:32]

qqnorm(est.b, ylab="Clutch Effects", xlab="Normal Scores")

**# MODEL 3 RANDOM CLUTCH INTERCEPTS**

**# Shrinkage Prior on Clutch Variance**

cat("model {for (h in 1:n) {for (s in 1:T1) {

L.tem[h,s] <- pow(L[h,s],q[s])

w[h,s] ~ dunif(a1[h,s],b1[h,s])

a1[h,s] <- -1/L.tem[h,s];

b1[h,s] <- 1/L.tem[h,s]

LL[h,s] <- log(L[h,s])

pi[h,s] <- phi(beta1[s]+beta2[s]\*x[h]+b[clutch[h],s])

# log-likelihood

log(L[h,s]) <- y[h]\*log(pi[h,s])+(1-y[h])\*log(1-pi[h,s])}}

# priors over path (1,...,T+1)

# regression parameters

est.beta[1] <- beta1[T1]

est.beta[2] <- beta2[T1]

for (s in 1:T1) {beta1[s] ~ dnorm(0,1)

beta2[s] ~ dnorm(0,1)}

# clutch random effects

for (j in 1:J) { est.b[j] <- b[j,T1]

# assess significance of clutch effects

pos.b[j] <- step(est.b[j])

for (s in 1:T1) {b[j,s] ~ dnorm(0,tau.b[s])}}

# random effect variance

est.s2 <- 1/tau.b[T1]

# shrinkage prior on clutch variance

# P known

# for (s in 1:T1) {U[s] ~ dunif(0,1)

# sigma2.b[s] <- (1-pow(U[s],0.5))/pow(U[s],0.5)

# tau.b[s] <- 1/sigma2.b[s]}

# P unknown

for (s in 1:T1) {U[s] ~ dunif(0,1)

sigma2.b[s] <- (1-pow(U[s],1/P[s]))/pow(U[s],1/P[s])

tau.b[s] <- 1/sigma2.b[s]

P1[s] ~ dgamma(0.1,0.1)

P[s] <- P1[s]+1}

est.P <- P[T1]

# path sampling calculations

for (s in 1:T1) { q[s] <- pow(a[s],path.pow)

expLL[s] <- sum(LL[1:n,s]) }

for (s in 1:T) { mc[s] <- (q[s+1]-q[s])\*(expLL[s+1]+expLL[s])\*0.5 }

logML <- sum(mc[])}

", file="model3.jag")

inits1 = list(b=matrix(0,J,T1), beta1=rep(-2.9,T1), beta2=rep(0.4,T1))

inits2 = list(b=matrix(0,J,T1), beta1=rep(-2.8,T1), beta2=rep(0.35,T1))

inits=list(inits1,inits2)

pars = c("logML","est.b","est.s2","est.beta","pos.b","est.P")

R3 = autojags(D, inits, pars,model.file="model3.jag",2,iter.increment=500, n.burnin=100,Rhat.limit=1.1, max.iter=5000, seed=1234)

R3$summary

ML=BF=c()

ML[1]=R1$summary[1]

ML[2]=R2$summary[1]

ML[3]=R3$summary[1]

**# Bayes Factor according to different priors on clutch effects**

BF[1]=exp(ML[1]-ML[2])

BF[2] =exp(ML[1]-ML[3])

#

**# COMPARISONS USING RSTAN AND BRIDGE-SAMPLING**

#

**# Fixed Effects**

M1\_code <-

"data {

int<lower = 1> N;

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

real<lower = 0> x[N];

}

parameters {

real alpha0\_n01;

real alpha1\_n01;

}

transformed parameters {

real alpha0 = sqrt(10.0) \* alpha0\_n01;

real alpha1 = sqrt(10.0) \* alpha1\_n01;

}

model {

target += normal\_lpdf(alpha0\_n01 | 0, 1); // prior

target += normal\_lpdf(alpha1\_n01 | 0, 1); // prior

for (i in 1:N) { // likelihood

target += bernoulli\_lpmf(y[i] | Phi(alpha0 + alpha1 \* x[i]));

}

}"

**### M2, random intercepts**

M2\_code <-

"data {

int<lower = 1> N;

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 alpha0\_n01;

real alpha1\_n01;

vector[C] b\_n01;

real<lower = 0> sigma2;

}

transformed parameters {

vector[C] b;

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

real alpha0 = sqrt(10.0) \* alpha0\_n01;

real alpha1 = sqrt(10.0) \* alpha1\_n01;

b = sigma \* b\_n01;

}

model {

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

target += normal\_lpdf(alpha0\_n01 | 0, 1); // prior

target += normal\_lpdf(alpha1\_n01 | 0, 1); // prior

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

for (i in 1:N) { // likelihood

target += bernoulli\_lpmf(y[i] | Phi(alpha0 + alpha1 \* x[i] +

b[clutch[i]]));

}

}"

**# M4, random clutch slopes**

M4\_code <-

"data {

int<lower = 1> N;

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 alpha0\_n01;

real alpha1\_n01;

vector[C] b\_n01;

real<lower = 0> sigma2;

}

transformed parameters {

vector[C] b;

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

real alpha0 = sqrt(10.0) \* alpha0\_n01;

real alpha1 = sqrt(10.0) \* alpha1\_n01;

b = sigma \* b\_n01;

}

model {

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

target += normal\_lpdf(alpha0\_n01 | 0, 1); // prior

target += normal\_lpdf(alpha1\_n01 | 0, 1); // prior

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

for (i in 1:N) { // likelihood

target += bernoulli\_lpmf(y[i] | Phi(alpha0 + (alpha1+b[clutch[i]])\* x[i] ));

}

}"

**# Estimation**

set.seed(1)

data0 = list(y = turtles$y, x = turtles$x, N = 244)

data1=list(y = turtles$y,x = turtles$x,N =244,C = max(turtles$clutch),clutch = turtles$clutch)

fit\_M1 <- stan(model\_code = M1\_code,data=data0, iter = 5000, warmup = 500, chains = 2, seed = 1)

fit\_M2 <- stan(model\_code = M2\_code,data=data1, iter = 5000, warmup = 500,chains = 2, seed = 1)

fit\_M4 <- stan(model\_code = M4\_code,data=data1, iter = 5000, warmup = 500, chains = 2, seed = 1)

**# Log marginal likelihoods**

set.seed(1)

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

bridge\_M2 <- bridge\_sampler(fit\_M2)

bridge\_M4 <- bridge\_sampler(fit\_M4)

**# Approximate percentage errors**

error\_measures(bridge\_M1)$percentage

error\_measures(bridge\_M2)$percentage

error\_measures(bridge\_M4)$percentage

**# Bayes Factors**

bf(bridge\_M1, bridge\_M2)

bf(bridge\_M1, bridge\_M4)