library(bridgesampling)

options(scipen=999)

data("turtles")

y=turtles$y

x=turtles$x

C=turtles$clutch

N = length(y)

J = length(unique(C))

# posterior density function

f = function(beta,alpha,tau,e) { sig = 1/sqrt(tau)

# survival model

for (i in 1:N){ p[i] = pnorm(alpha+beta\*x[i]+e[C[i]])

LL[i] = y[i]\*log(p[i])+(1-y[i])\*log(1-p[i])}

# prior ordinates

logpr[1] = -0.5\*alpha^2/10

logpr[2] = -0.5\*beta^2/10

logpr[3] = -0.001\*tau

for (j in 1:J){ LLr[j] = -0.5\*e[j]^2/sig^2-log(sig)}

# log-posterior

f = sum(LL[1:N])+sum(LLr[1:J])+sum(logpr[1:3])}

# MCMC settings

T = 5000

# warm up

B =T/10

# accumulate M-H rejections for hyperparameters

k1 = 0; k2 = 0; k3 =0

# gamma parameter for precision updates

kappa=100

# uniform samples for use in hyperparameter updates

U1 = U2 = U3 = log(runif(T))

# define arrays

alpha = numeric(T); beta = numeric(T); tau = numeric(T); logpr = numeric(3)

s = numeric(T); p = numeric(N); e = numeric(J); LL = numeric(N);

LLr = numeric(J); ec = matrix(0,T,J); en = matrix(0,T,J);

kran = numeric(J)

# initial parameter values

beta[1]= 0.35; alpha[1]= -2.6; tau[1]= 5; for (j in 1:J) {ec[1,j]= 0; kran[j]= 0}

# Main loop

# update beta

for (t in 2:T) {bstar = beta[t-1]+0.05\*rnorm(1,0,1)

tn = f(bstar,alpha[t-1],tau[t-1],ec[t-1,]); tf = f(beta[t-1],alpha[t-1],tau[t-1],ec[t-1,])

if (U1[t] <= tn-tf) beta[t] = bstar

else {beta[t] = beta[t-1]; k1 = k1+1 }

# update intercept

astar = alpha[t-1]+0.5\*rnorm(1,0,1)

tn = f(beta[t],astar,tau[t-1],ec[t-1,]); tf = f(beta[t],alpha[t-1],tau[t-1] ,ec[t-1,])

if (U2[t] <= tn-tf) alpha[t] = astar

else {alpha[t] = alpha[t-1]; k2 = k2+1}

# update precision

taustar = rgamma(1,kappa,kappa/tau[t-1])

s[t-1] = 1/sqrt(tau[t-1])

tn = f(beta[t],alpha[t],taustar,ec[t-1,])+log(dgamma(tau[t-1],kappa,kappa/taustar))

tc = f(beta[t],alpha[t],tau[t-1],ec[t-1,])+log(dgamma(taustar,kappa,kappa/tau[t-1]))

if (U3[t] <= tn-tf) tau[t] = taustar

else {tau[t] = tau[t-1]; k3 = k3+1}

# update cluster effects

for (j in 1:J) { en[j] = ec[t-1,j]

ec[t,j] = ec[t-1,j]}

for (j in 1:J) { en[j] = ec[t-1,j]+rnorm(1,0,1)

tn = f(beta[t],alpha[t],tau[t],en[]); tf = f(beta[t],alpha[t],tau[t],ec[t,])

if (log(runif(1)) <= tn-tf) ec[t,j] = en[j]

else { en[j] = ec[t-1,j]

kran[j] = kran[j]+1}}}

# hyperparameter summaries

quantile(alpha[B:T], probs=c(.025,0.5,0.975))

quantile(beta[B:T], probs=c(.025,0.5,0.975))

quantile(tau[B:T], probs=c(.025,0.5,0.975))

quantile(s[B:T], probs=c(.025,0.5,0.975))

# random effects posterior means and quantiles

eff.mdn = apply(ec[B:T,], 2, quantile, probs = c(0.50))

eff.q975=apply(ec[B:T,], 2, quantile, probs = c(0.975))

eff.q025=apply(ec[B:T,], 2, quantile, probs = c(0.025))

eff.q90=apply(ec[B:T,], 2, quantile, probs = c(0.90))

eff.q10=apply(ec[B:T,], 2, quantile, probs = c(0.10))

# number of significant 80% credible intervals for random effects

sum(eff.q90>0 & eff.q10 >0)+ sum(eff.q90<0 & eff.q10 <0)

# acceptance rates for hyperparameters (beta, alpha, tau.b)

1-k1/T; 1-k2/T; 1-k3/T

# acceptance rates for cluster effects

1-kran/T