HW-6

Monday, November 28, 2022 7:37 PM

3. $Y_1, Y_2 \sim$ Bernoulli

a) Given: X – discrete random variable
Y_1, Y_2 – conditionally iid given x

$$g(x) = P(Y_1=1|X=x) = P(Y_2=1|X=x) \quad – ①$$

i) Given:

$$P(Y_1=1, Y_2=1) = \sum_x P(Y_1=1, Y_2=1 | X=x)\, P(X=x) \quad \text{[By law of total probability]}$$

Show that:

$P(Y_1=1, Y_2=1) = 0$ implies $g(x)=0 \quad \forall x$ s.t $P(X=x)>0$

Solution:

$$\begin{aligned} P(Y_1=1, Y_2=1) &= \sum_x P(Y_1=1, Y_2=1 | X=x).P(X=x) \quad [\because \text{Law of total probability}] \\ 0 &= \sum_x P(Y_1=1|X=x).P(Y_2=1|X=x).P(X=x) \quad [Y_1, Y_2 \text{ conditionally iid given } X] \\ 0 &= \sum_x g(x).g(x).P(X=x) \quad [Y_1, Y_2 \text{ identically dist. given } X] \\ 0 &= \sum_x [g(x)]^2.P(X=x) \end{aligned}$$

For all x s.t $P(X=x)>0$, the above expression equals 0 when

$$\sum_x [g(x)]^2 \cdot \underset{\text{positive}}{\underset{\Downarrow}{P(X=x)}} = 0$$

$\Rightarrow [g(x)]^2 = 0$ for all x s.t. $P(X=x)>0$

$\Rightarrow g(x) = 0 \quad \forall x$ s.t $P(X=x)>0$

ii) Show that:

$g(x)=0$ for all x s.t. $P(X=x)>0$ implies

$P(Y_1=1) = P(Y_2=1) = 0$

Solution:

$$\begin{aligned} P(Y_1=1) &= \sum_x P(Y_1=1|X=x).P(X=x) \\ &= \sum_x g(x).P(X=x) \quad [\because g(x)=0] \\ &= \sum_x 0 \; P(X=x) \quad \Rightarrow P(Y_1=1)=0 \end{aligned}$$

Similarly,

$$\begin{aligned} P(Y_2=1) &= \sum_x P(Y_2=1|X=x).P(X=x) \\ &= \sum_x g(x).P(X=x) \\ &= \sum_x 0 \; P(X=x) \quad \Rightarrow P(Y_2=1)=0 \end{aligned}$$

b) Given: $P(Y_1=1, Y_2=0) = P(Y_1=0, Y_2=1) = 1/2 \quad – ②$

i) Y_1 & Y_2 are exchangeable because reordering the indices do not change their joint distribution i.e.
$P(Y_1=1, Y_2=0) = P(Y_1=0, Y_2=1) = 1/2$
where $Y_i \sim$ Bernoulli, $i = \{0,1\}$

ii) **Show there cannot exist any discrete random variable X s.t. Y_1, Y_2 are conditionally iid given x.**

From ②, $P(Y_1=1, Y_2=1) = 0$

Let $g(x) = P(Y_1=1|X=x) = P(Y_2=1|X=x)$ [$\because Y_1, Y_2$ are exchangeable]

From a) i), $P(Y_1=1, Y_2=1) = 0 \Rightarrow g(x)=0 \quad \forall x$ s.t $P(X=x)>0$

From a) ii), $g(x)=0 \quad \forall x$ s.t. $P(X=x)>0 \Rightarrow P(Y_1=1) = P(Y_2=1) = 0$

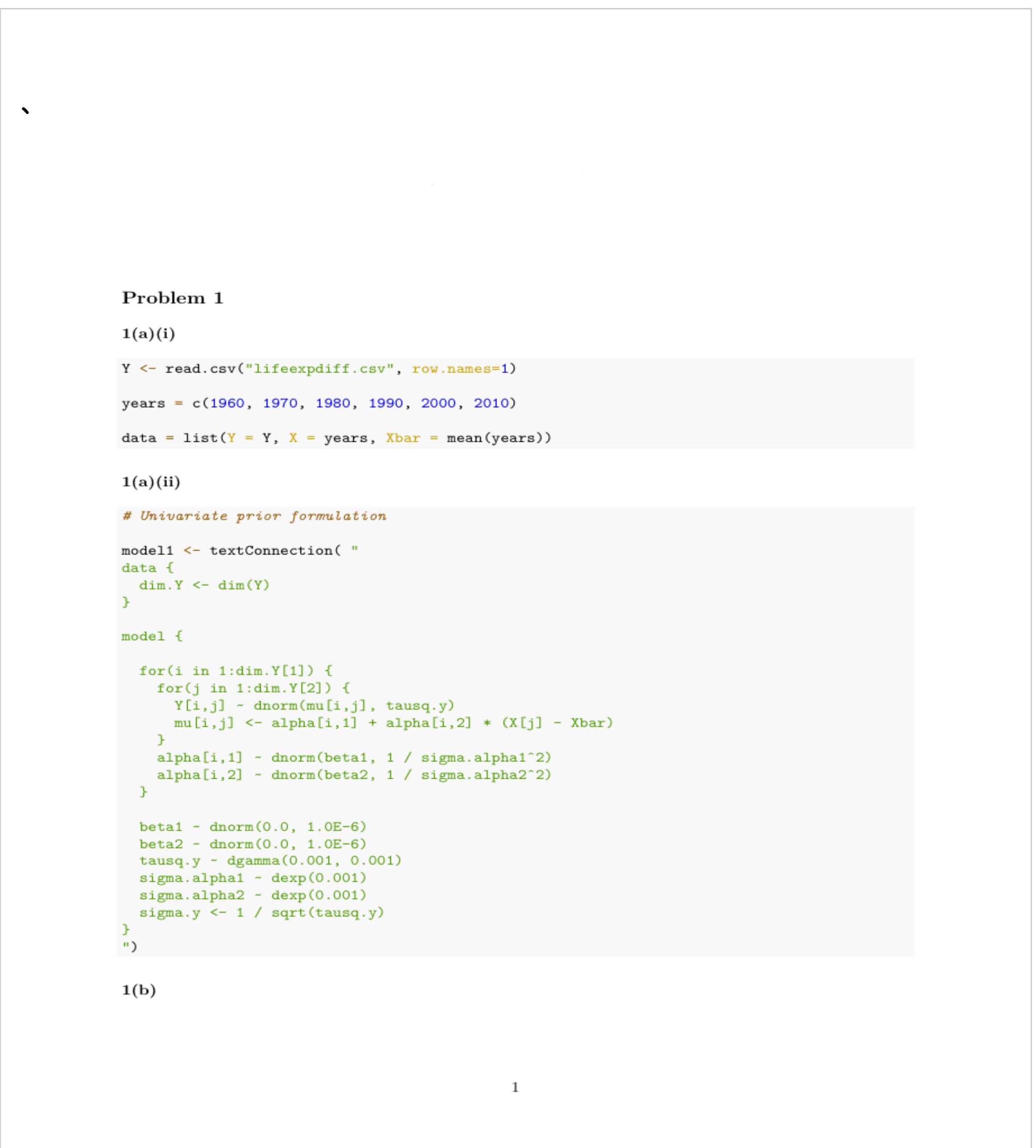
$$\begin{aligned} P(Y_1=1, Y_2=0) &= \sum_x P(Y_1=1|X=x).P(Y_2=0|X=x).P(X=x) \quad [\because Y_1, Y_2 \text{ are conditionally iid}] \\ &= \sum_x 0 \cdot P(Y_2=0|X=x)\,P(X=x) \end{aligned}$$

$P(Y_1=1, Y_2=0) = 0$ CONTRADICTION!! ③

From ②, we know $P(Y_1=1, Y_2=0) = 1/2$ but ③ gives a contraction.

$\Rightarrow$ So, we can't have X s.t. $\forall x$ $P(X=x)>0$

$\Rightarrow$ No such random variable X can exist.



Problem 1

1(a)(i)

```
Y <- read.csv("lifeexpdiff.csv", row.names=1)

years = c(1960, 1970, 1980, 1990, 2000, 2010)

data = list(Y = Y, X = years, Xbar = mean(years))
```

1(a)(ii)

```
# Univariate prior formulation

model1 <- textConnection( "
data {
  dim.Y <- dim(Y)
}

model {

  for(i in 1:dim.Y[1]) {
    for(j in 1:dim.Y[2]) {
      Y[i,j] ~ dnorm(mu[i,j], tausq.y)
      mu[i,j] <- alpha[i,1] + alpha[i,2] * (X[j] - Xbar)
    }
    alpha[i,1] ~ dnorm(beta1, 1 / sigma.alpha1^2)
    alpha[i,2] ~ dnorm(beta2, 1 / sigma.alpha2^2)
  }

  beta1 ~ dnorm(0.0, 1.0E-6)
  beta2 ~ dnorm(0.0, 1.0E-6)
  tausq.y ~ dgamma(0.001, 0.001)
  sigma.alpha1 ~ dexp(0.001)
  sigma.alpha2 ~ dexp(0.001)
  sigma.y <- 1 / sqrt(tausq.y)
}
")
```

1(b)

```
library(rjags)

## Loading required package: coda

## Linked to JAGS 4.3.1

## Loaded modules: basemod,bugs

inits = list(list(tausq.y=1, beta1=0, beta2=0,
                  sigma.alpha1=1, sigma.alpha2=1),
        list(tausq.y=0.1, beta1=10, beta2=10,
             sigma.alpha1=0.1, sigma.alpha2=0.1),
        list(tausq.y=0.01, beta1=-10, beta2=-10,
             sigma.alpha1=10, sigma.alpha2=10))


uni.model = jags.model(model1, data, inits, n.chains=3)

## Compiling data graph
##    Resolving undeclared variables
##    Allocating nodes
##    Initializing
##    Reading data back into data table
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 300
##    Unobserved stochastic nodes: 105
##    Total graph size: 1031
##
## Initializing model

uni.model.run <- coda.samples(uni.model, c("beta1","beta2","sigma.y","sigma.alpha1","sigma.alpha2
                  n.iter=100000, n.burnin = 1000)

#summary statistic table
summary(window(uni.model.run, 400))

##
## Iterations = 1001:101000
## Thinning interval = 1
## Number of chains = 3
## Sample size per chain = 1e+05
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                   Mean       SD  Naive SE Time-series SE
## beta1         0.116953 0.190292 3.474e-04      3.502e-04
## beta2        -0.005431 0.003676 6.712e-06      7.595e-06
```

```
## sigma.alpha1  1.329254 0.141000 2.574e-04      3.754e-04
## sigma.alpha2  0.024224 0.002891 5.278e-06      8.571e-06
## sigma.y       0.382485 0.019220 3.509e-05      5.022e-05
##
## 2. Quantiles for each variable:
##
##                  2.5%       25%       50%       75%   97.5%
## beta1        -0.25814 -0.009938  0.117273  0.24353 0.49150
## beta2        -0.01268 -0.007877 -0.005426 -0.00299 0.00181
## sigma.alpha1  1.08731  1.229695  1.317652  1.41553 1.63936
## sigma.alpha2  0.01919  0.022198  0.024004  0.02601 0.03051
## sigma.y       0.34700  0.369147  0.381696  0.39495 0.42238
```

1(c)

```
# 95% confidence interval for beta1
quantile(unlist(uni.model.run[ , "beta1"]), c(0.025, 0.975))

##       2.5%      97.5%
## -0.2581416  0.4914958

# 95% confidence interval for beta2
quantile(unlist(uni.model.run[ , "beta2"]), c(0.025, 0.975))

##         2.5%        97.5%
## -0.012683478  0.001809773
```

Considering a flat prior, we can see that the posterior distribution for beta has same shape as the likelihood of the prior. This implies that as the likelihood has a 0 mean, we have 0 in the confidence intervals as well.

Problem 2

2(a)

```
data_new =  list(Y = Y, X = years, Xbar = mean(years),
                 Omega0 = rbind(c(100, 0),c(0, 0.1)),
                 mu0 = c(0,0),
                 Sigma.inv = rbind(c(1.0E-6, 0),c(0, 1.0E-6)))

# Bivariate prior formulation
model2 <- textConnection("
data {
  dim.Y <- dim(Y)
}

model {
  for(i in 1:dim.Y[1]) {
```

```
    for(j in 1:dim.Y[2]) {
      Y[i,j] ~ dnorm(mu[i,j], tausq.y)
      mu[i,j] <- alpha[i,1] + alpha[i,2] * (X[j] - Xbar)
    }

    alpha[i,1:2] ~ dmnorm(beta, Omega.inv)
  }

  tausq.y ~ dgamma(0.001, 0.001)
  beta ~ dmnorm(mu0, Sigma.inv)
  Omega.inv ~ dwish(2*Omega0, 2)
  Omega <- inverse(Omega.inv)
  sigma.y <- 1 / sqrt(tausq.y)

  rho <- Omega[1,2] / sqrt(Omega[1,1] * Omega[2,2])
  pb <- rho > 0
}
")
```

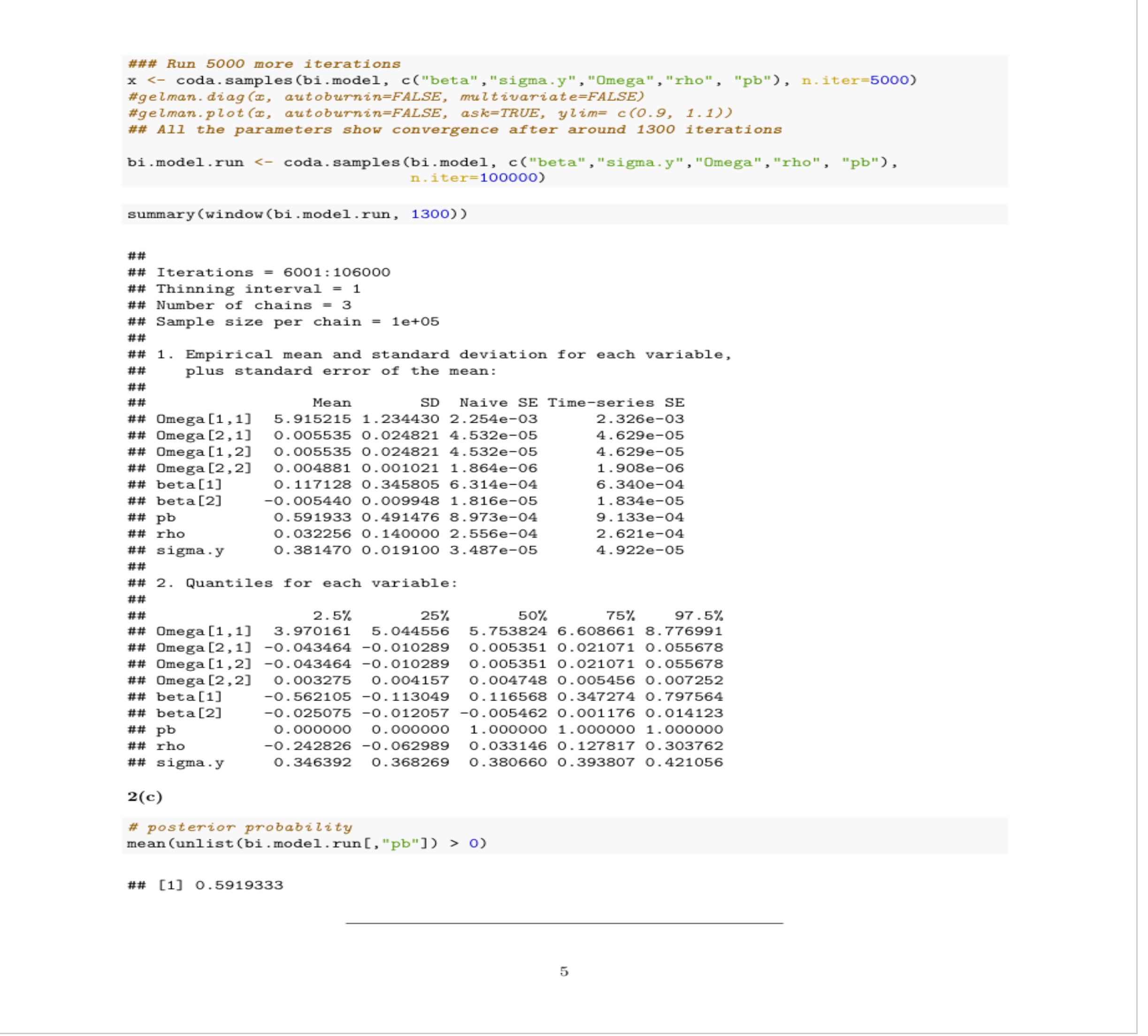
2(b)

```
inits_new <- list(list(tausq.y=1, beta=c(0,0),
                       Omega.inv=diag(2)),
                  list(tausq.y=10, beta=c(10,10),
                       Omega.inv=10*diag(2)),
                  list(tausq.y=100, beta=c(-10,-10),
                       Omega.inv=100*diag(2)))

bi.model<- jags.model(model2, data_new, inits_new, n.chains=3)

## Compiling data graph
##    Resolving undeclared variables
##    Allocating nodes
##    Initializing
##    Reading data back into data table
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 300
##    Unobserved stochastic nodes: 53
##    Total graph size: 1096
##
## Initializing model

x = coda.samples(bi.model, c("beta","sigma.y","Omega" ,"rho", "pb"), n.iter=1000)
### Assess convergence
#autocorr.plot(x[1], ask=TRUE)
#gelman.diag(x, autoburnin=FALSE, multivariate=FALSE)
#gelman.plot(x, autoburnin=FALSE, ask=TRUE, ylim= c(0.9, 1.1))
## Since the plots don't show convergence in case of some parameters, we run it longer.
```



```
### Run 5000 more iterations
x <- coda.samples(bi.model, c("beta","sigma.y","Omega","rho", "pb"), n.iter=5000)
#gelman.diag(x, autoburnin=FALSE, multivariate=FALSE)
#gelman.plot(x, autoburnin=FALSE, ask=TRUE, ylim= c(0.9, 1.1))
## All the parameters show convergence after around 1300 iterations

bi.model.run <- coda.samples(bi.model, c("beta","sigma.y","Omega","rho", "pb"),
                             n.iter=100000)

summary(window(bi.model.run, 1300))

##
## Iterations = 6001:106000
## Thinning interval = 1
## Number of chains = 3
## Sample size per chain = 1e+05
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                  Mean       SD  Naive SE Time-series SE
## Omega[1,1]   5.915215 1.234430 2.254e-03      2.326e-03
## Omega[2,1]   0.005535 0.024821 4.532e-05      4.629e-05
## Omega[1,2]   0.005535 0.024821 4.532e-05      4.629e-05
## Omega[2,2]   0.004881 0.001021 1.864e-06      1.908e-06
## beta[1]      0.117128 0.345805 6.314e-04      6.340e-04
## beta[2]     -0.005440 0.009948 1.816e-05      1.834e-05
## pb           0.591933 0.491476 8.973e-04      9.133e-04
## rho          0.032256 0.140000 2.556e-04      2.621e-04
## sigma.y      0.381470 0.019100 3.487e-05      4.922e-05
##
## 2. Quantiles for each variable:
##
##                  2.5%       25%       50%      75%    97.5%
## Omega[1,1]   3.970161  5.044556  5.753824 6.608661 8.776991
## Omega[2,1]  -0.043464 -0.010289  0.005351 0.021071 0.055678
## Omega[1,2]  -0.043464 -0.010289  0.005351 0.021071 0.055678
## Omega[2,2]   0.003275  0.004157  0.004748 0.005456 0.007252
## beta[1]     -0.562105 -0.113049  0.116568 0.347274 0.797564
## beta[2]     -0.025075 -0.012057 -0.005462 0.001176 0.014123
## pb           0.000000  0.000000  1.000000 1.000000 1.000000
## rho         -0.242826 -0.062989  0.033146 0.127817 0.303762
## sigma.y      0.346392  0.368269  0.380660 0.393807 0.421056
```

2(c)

```
# posterior probability
mean(unlist(bi.model.run[,"pb"]) > 0)

## [1] 0.5919333
```
