Homework 11 Solution

Math 315, Fall 2019

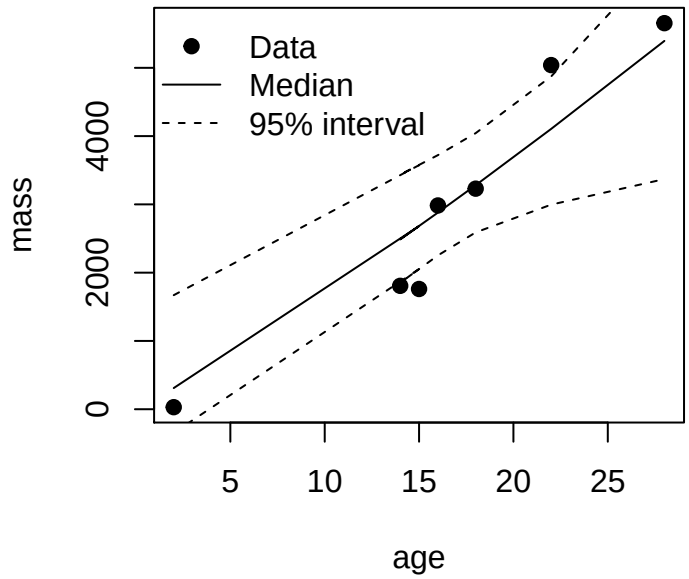
Exercise 3.15

(a)

Note: I used three chains to help investigate convergence/mixing. The book didn't specify how many chains you should used, so multiple answers are accepted. In addition, multiple plots comparing μ_i and the data are possible. I show one example below.

```
# Load rjags
library(rjags)

# Data
mass <- c(29.9, 1761, 1807, 2984, 3230, 5040, 5654)
age  <- c(2, 15, 14, 16, 18, 22, 28)
n    <- length(age)
data <- list(mass = mass, age = age, n = n)

# JAGS model specification
model_string <- textConnection("model{
   # Likelihood
    for(i in 1:n){
      mass[i]  ~ dnorm(mu[i],tau)
      mu[i]   <- theta[1] + theta[2]*pow(age[i],theta[3])
    }

   # Priors
    theta[1] ~ dnorm(0, 0.00001)
    theta[2] ~ dunif(0, 20000)
    theta[3] ~ dnorm(0,1)
    tau      ~ dgamma(0.01,0.01)
}")

# Compile the model
 model <- jags.model(model_string, data = data, n.chains = 3, quiet=TRUE)

# Burnon
update(model, 10000, progress.bar="none")

# Drawing posterior samples for mu and theta
params  <- c("mu","theta")
samples <- coda.samples(model, variable.names = params,
                        n.iter = 10000, progress.bar = "none")

# Plotting the fit (posterior median)
post_sum <- summary(samples)
stats    <- post_sum$statistics
quants   <- post_sum$quantiles

plot(age, mass, pch = 19)
```

```
lines(age, stats[1:n,1])         # Posterior medians
lines(age, quants[1:n,1],lty=2)  # Posterior 95% interval
lines(age, quants[1:n,5],lty=2)
legend("topleft", c("Data","Median","95% interval"),
       lty = c(NA, 1, 2), pch = c(19, NA, NA), bty = "n")
```

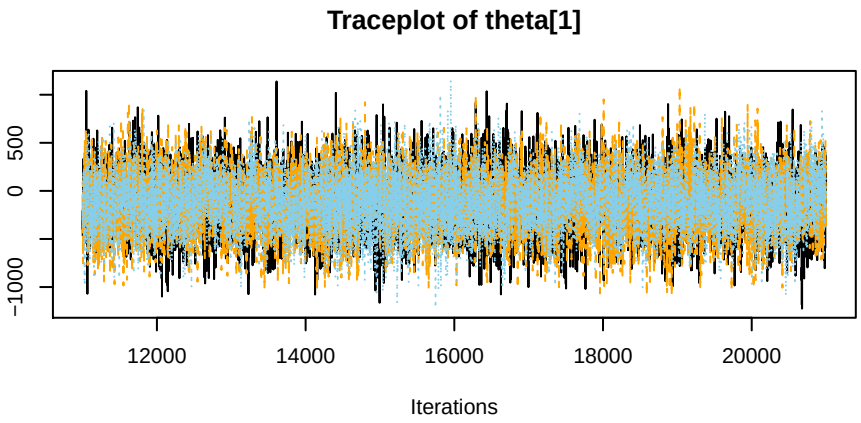


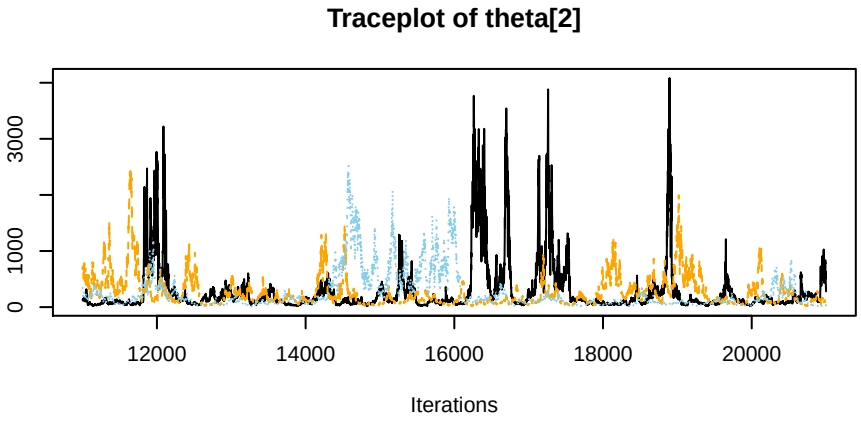
The model fits reasonably well, though it does underpredict the mass for all ages less than 15.

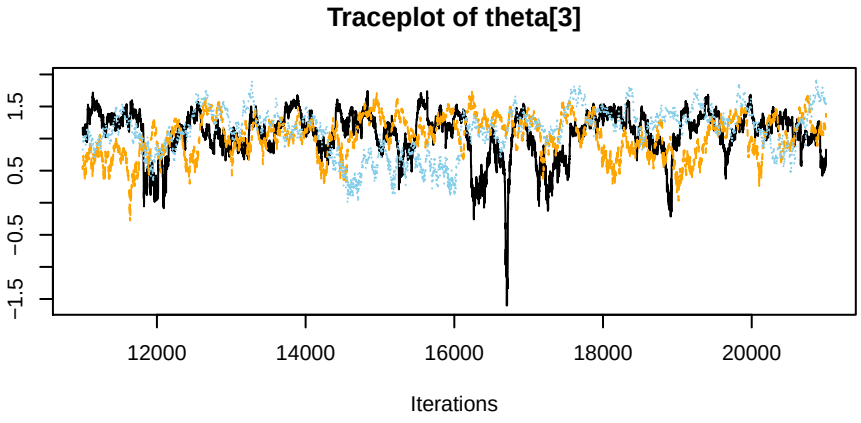
(b)

Convergence is fine for θ_1, but poor for θ_2 and θ_3. You can see this by (1) the low effective sample sizes, (2) the Gelman-Rubin statistic for θ_3, and (3) trace plots. Note: Since I ran multiple chains I used the Gelman-Rubin diagnostic as opposed to the Geweke diagnostic.

```
# traceplots
par(mfrow = c(3, 1))
theta1_samples <- list(samples[[1]][,"theta[1]"], samples[[2]][,"theta[1]"], samples[[3]][,"theta[1]"])
theta2_samples <- list(samples[[1]][,"theta[2]"], samples[[2]][,"theta[2]"], samples[[3]][,"theta[2]"])
theta3_samples <- list(samples[[1]][,"theta[3]"], samples[[2]][,"theta[3]"], samples[[3]][,"theta[3]"])
traceplot(theta1_samples, col = c("black", "orange", "skyblue"), main = "Traceplot of theta[1]")
traceplot(theta2_samples, col = c("black", "orange", "skyblue"), main = "Traceplot of theta[2]")
traceplot(theta3_samples, col = c("black", "orange", "skyblue"), main = "Traceplot of theta[3]")
```








```
# Convergence diagnostics
effectiveSize(samples)
```

```
##      mu[1]      mu[2]      mu[3]      mu[4]      mu[5]      mu[6]
##  243.28108  386.57526  279.87628  632.09256 2539.78656 1502.66896
##      mu[7]   theta[1]   theta[2]   theta[3]
##  245.91131 6891.61513  164.56311   95.85004
```

```
gelman.diag(samples, multivariate = F)
```

```
## Potential scale reduction factors:
```

```
##
##          Point est. Upper C.I.
## mu[1]          1.02       1.02
## mu[2]          1.01       1.02
## mu[3]          1.01       1.02
## mu[4]          1.01       1.02
## mu[5]          1.00       1.01
## mu[6]          1.00       1.01
## mu[7]          1.01       1.02
## theta[1]       1.00       1.00
## theta[2]       1.06       1.07
## theta[3]       1.02       1.04
```

(c)

- One approach to improving convergence is to simply run the chains longer. This will likely work here because convergence isn't hopeless and the code is fast.
- A second approach is to simplify the model. The log-linear model

$$\log(mass) = \beta_0 + \beta_1 age + \varepsilon$$

  is one possibility.
- Finally, the posterior draws of θ_2 and θ_3 are highly correlated (check out the `crosscorr()` function), so updating them in a block might improve convergence.

Exercise 3.16

(a)

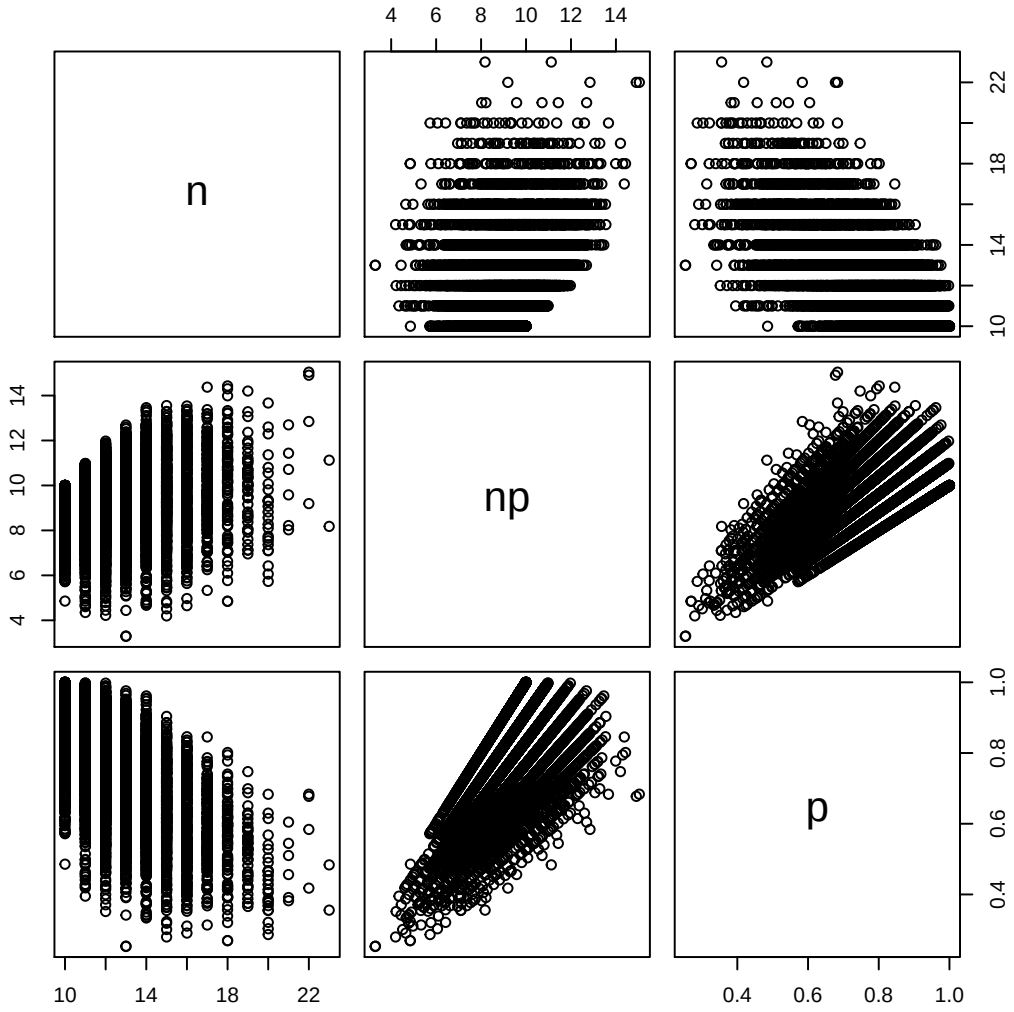
There are two parameters and only one observation, so it will be impossible to identify both.

(b)

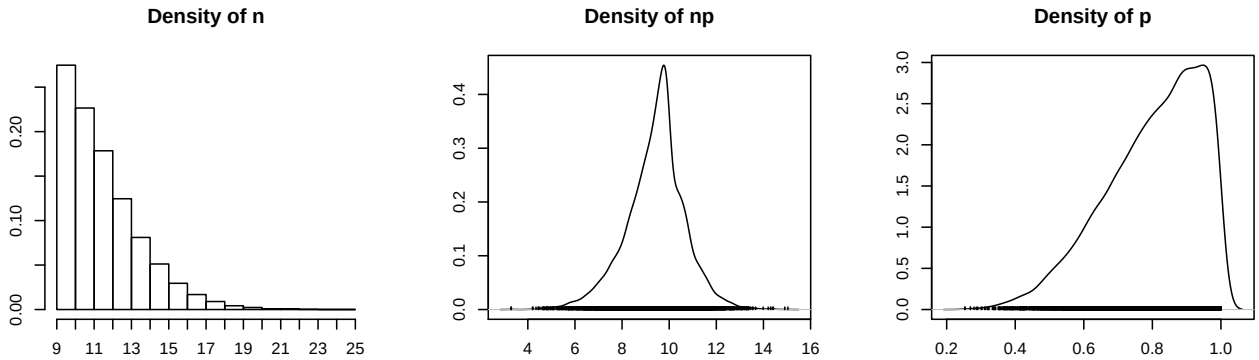
As shown below, convergence isn't terrible but the posterior distributions are wide. Convergence is better for np than n or p. This is because $E(Y) = np$, so the product is identified.

```
lambda <- 10
a      <- 1
b      <- 1
Y      <- 10
data   <- list(Y = Y, a = a, b = b, lambda = lambda)

model_string <- textConnection("model{
   Y   ~ dbin(p,n)
   n   ~ dpois(lambda)
   p   ~ dbeta(a,b)
   np <- n*p
 }")

model <- jags.model(model_string,data = data, n.chains=2,quiet=TRUE)
update(model, 10000, progress.bar = "none")
params  <- c("n", "p", "np")
```

```
samples <- coda.samples(model,
                        variable.names = params,
                        n.iter = 10000, progress.bar="none")

pairs(as.matrix(samples[[1]]))
```




```
par(mfrow = c(1, 3))
densplot(samples)
```



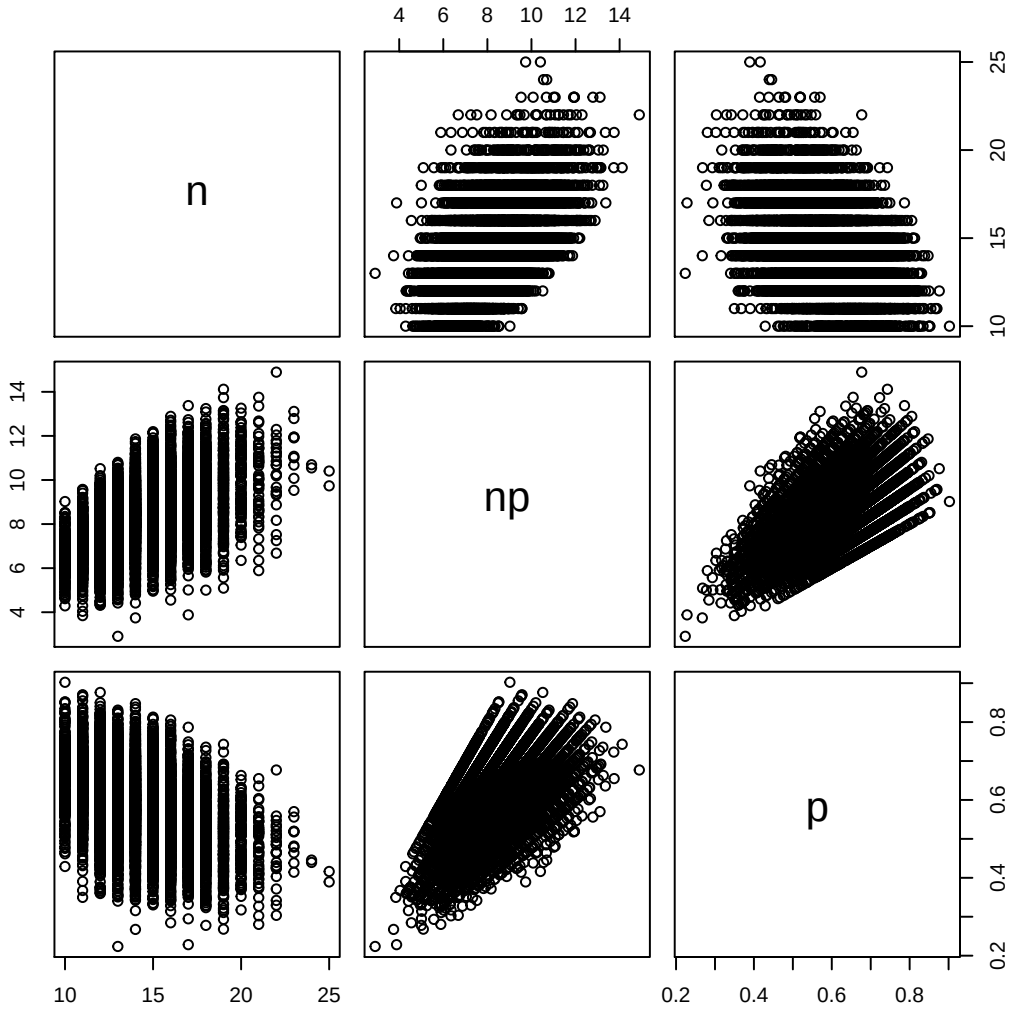
```
effectiveSize(samples)
```

```
##         n        np         p
##  6741.762 25686.485  6491.955
```

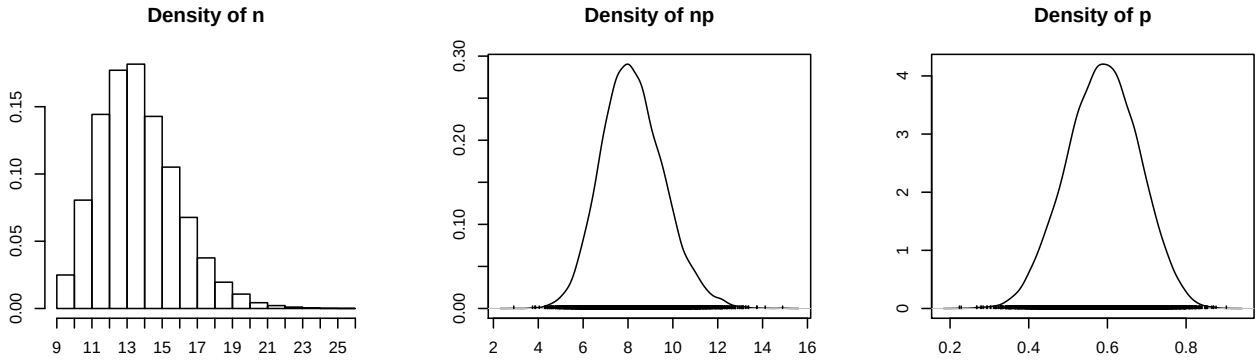
(c)

The tighter prior for p improves convergence and reduces posterior uncertainty for both parameters.

```
lambda <- 10
a      <- 10
b      <- 10
Y      <- 10
data   <- list(Y = Y, a = a, b = b, lambda = lambda)

model_string <- textConnection("model{
   Y   ~ dbin(p,n)
   n   ~ dpois(lambda)
   p   ~ dbeta(a,b)
   np <- n*p
 }")

model <- jags.model(model_string,data = data, n.chains=2,quiet=TRUE)
update(model, 10000, progress.bar = "none")
params  <- c("n", "p", "np")
samples <- coda.samples(model,
                        variable.names = params,
                        n.iter = 10000, progress.bar="none")
pairs(as.matrix(samples[[1]]))
```




```
par(mfrow = c(1, 3))
densplot(samples)
```




```
effectiveSize(samples)
```

```
##        n       np        p
## 14347.30 27158.76 13492.76
```

Exercise 4.2a-c

(a)

```
# Loading the data
library(MASS)
data(Boston)
X  <- scale(Boston[,1:13])
Y  <- as.vector(scale(Boston[,14]))


data <- list(n = length(Y), p = ncol(X), X = X, Y = Y)

# Specifying the model
model_string <- textConnection("model{

  # Likelihood
  for(i in 1:n){
    Y[i]   ~ dnorm(mu[i],tau)
    mu[i] <- alpha + inprod(X[i,],beta[])
  }
  for(j in 1:p){
    beta[j] ~ dnorm(0,0.01)
  }
  alpha ~  dnorm(0, 0.01)
  tau   ~  dgamma(0.1, 0.1)
}")

# Compile the model
model <- jags.model(model_string, data = data, n.chains = 3, quiet = TRUE)

# Burn-in
update(model, 10000, progress.bar = "none")

# Drawing posterior samples
params  <- "beta"
samples <- coda.samples(model,
                        variable.names =params,
                        n.iter=10000, progress.bar="none")

# Checking on convergence
gelman.diag(samples, multivariate = FALSE)
```

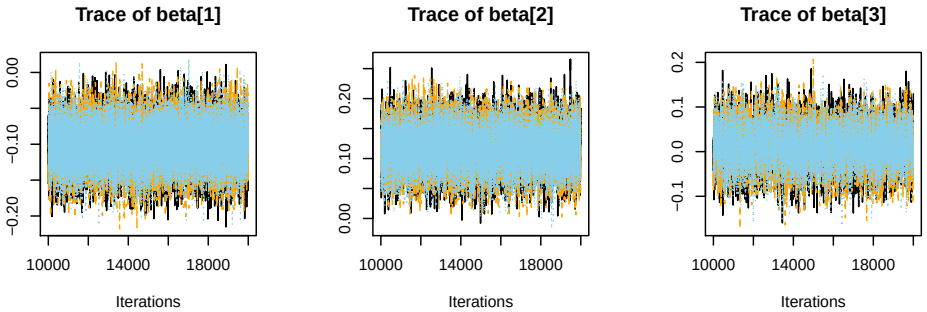
```
## Potential scale reduction factors:
##
##          Point est. Upper C.I.
## beta[1]           1          1
## beta[2]           1          1
## beta[3]           1          1
## beta[4]           1          1
## beta[5]           1          1
## beta[6]           1          1
## beta[7]           1          1
## beta[8]           1          1
```

```
## beta[9]           1          1
## beta[10]          1          1
## beta[11]          1          1
## beta[12]          1          1
## beta[13]          1          1
```

```
effectiveSize(samples)
```

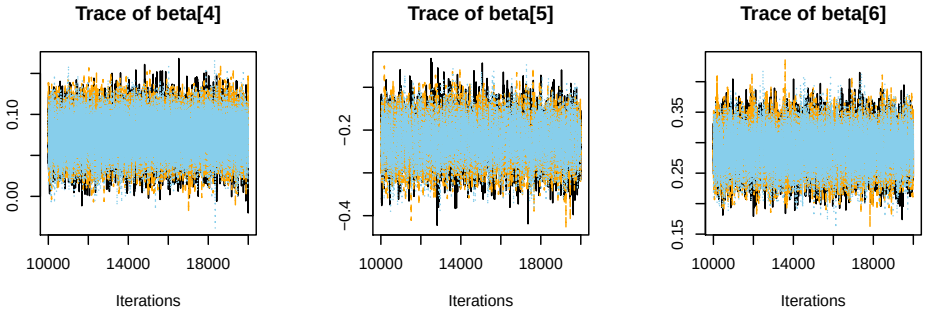
```
##   beta[1]   beta[2]   beta[3]   beta[4]   beta[5]   beta[6]   beta[7]
## 11699.484  6853.491  4424.827 23174.750  5560.062  7586.289  6537.425
##   beta[8]   beta[9]  beta[10]  beta[11]  beta[12]  beta[13]
##  5172.433  2211.897  1971.395 10430.912 17885.248  7265.462
```

```
par(mfrow = c(5, 3))
traceplot(samples, col = c("black", "orange", "skyblue"))
```


Trace of beta[1]
−0.20
−0.10
0.00
10000
14000
18000
Iterations

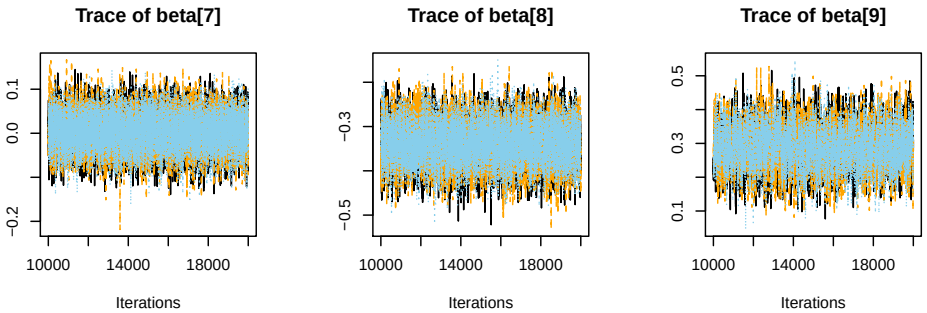
Trace of beta[2]
0.00
0.10
0.20
10000
14000
18000
Iterations

Trace of beta[3]
−0.1
0.0
0.1
0.2
10000
14000
18000
Iterations


Trace of beta[4]
0.00
0.10
10000
14000
18000
Iterations

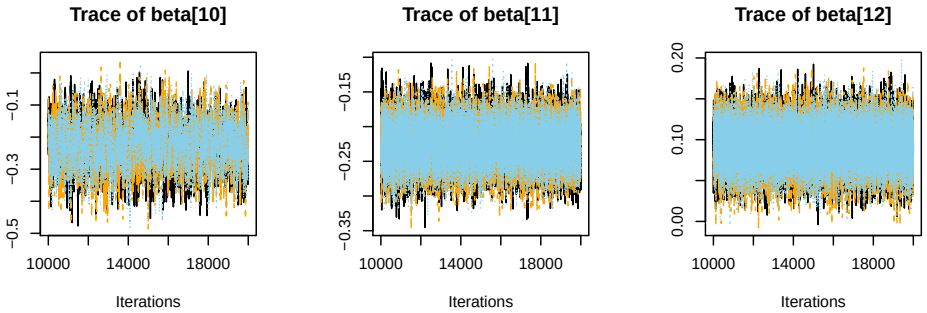
Trace of beta[5]
−0.4
−0.2
10000
14000
18000
Iterations

Trace of beta[6]
0.15
0.25
0.35
10000
14000
18000
Iterations


Trace of beta[7]
−0.2
0.0
0.1
10000
14000
18000
Iterations

Trace of beta[8]
−0.5
−0.3
10000
14000
18000
Iterations

Trace of beta[9]
0.1
0.3
0.5
10000
14000
18000
Iterations


Trace of beta[10]
−0.5
−0.3
−0.1
10000
14000
18000
Iterations

Trace of beta[11]
−0.35
−0.25
−0.15
10000
14000
18000
Iterations

Trace of beta[12]
0.00
0.10
0.20
10000
14000
18000
Iterations


Trace of beta[13]
−0.5
−0.4
−0.3
−0.2
10000
14000
18000
Iterations

The convergence of the chains looks great.

To summarize the posterior, you can create a table with the mean, SD, and credible interval for the model coefficients.

```
post_sum <- summary(samples)
post_table <- cbind(post_sum$statistics[,c("Mean", "SD")], post_sum$quantiles[,c("2.5%", "97.5%")])
knitr::kable(post_table, format = "markdown")
```

	Mean	SD	2.5%	97.5%
beta[1]	-0.1011645	0.0308571	-0.1613498	-0.0404568
beta[2]	0.1174704	0.0350805	0.0480040	0.1861261
beta[3]	0.0145130	0.0460001	-0.0768518	0.1040716
beta[4]	0.0741271	0.0239354	0.0272563	0.1210019
beta[5]	-0.2236976	0.0476704	-0.3176810	-0.1313414
beta[6]	0.2905648	0.0320805	0.2274735	0.3528359
beta[7]	0.0023967	0.0407105	-0.0773963	0.0812188
beta[8]	-0.3377510	0.0458236	-0.4278184	-0.2484777
beta[9]	0.2901261	0.0626433	0.1669071	0.4129341
beta[10]	-0.2259902	0.0692809	-0.3627464	-0.0904851
beta[11]	-0.2245383	0.0306843	-0.2849145	-0.1645911
beta[12]	0.0924542	0.0267989	0.0396149	0.1448903
beta[13]	-0.4075996	0.0395810	-0.4852981	-0.3303931

Alternatively, you could plot this information:

```
library(dplyr)
```

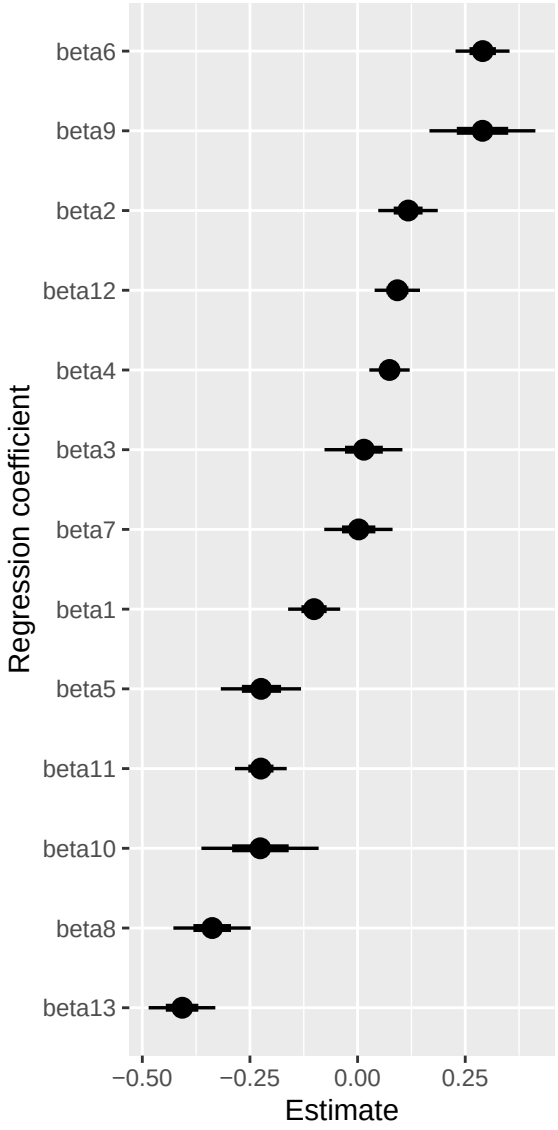
```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:MASS':
##
##     select
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(tidybayes)
library(forcats)
samples %>%
  gather_draws(beta[i]) %>%
  mutate(coef = paste0(.variable, i)) %>%
  ggplot(aes(y = fct_reorder(coef, .value), x = .value)) +
  stat_pointintervalh() +
  labs(x = "Estimate", y = "Regression coefficient")
```


beta6
beta9
beta2
beta12
beta4
beta3
beta7
beta1
beta5
beta11
beta10
beta8
beta13
Regression coefficient
−0.50
−0.25
0.00
0.25
Estimate

(b)

The results are nearly identical to the Bayesian analysis with uninformative priors, as expected. Of course, the interpretation of the intervals will differ.

```
classical_lm <- lm(Y ~ X)
classical_sum <- summary(classical_lm)$coef
knitr::kable(cbind(classical_sum[,1:2], confint(classical_lm)), format = "markdown", digits = 3)
```

	Estimate	Std. Error	2.5 %	97.5 %
(Intercept)	0.000	0.023	-0.045	0.045
Xcrim	-0.101	0.031	-0.161	-0.041
Xzn	0.118	0.035	0.049	0.186
Xindus	0.015	0.046	-0.075	0.105
Xchas	0.074	0.024	0.027	0.121
Xnox	-0.224	0.048	-0.318	-0.129
Xrm	0.291	0.032	0.228	0.354
Xage	0.002	0.040	-0.077	0.082
Xdis	-0.338	0.046	-0.428	-0.248
Xrad	0.290	0.063	0.166	0.413
Xtax	-0.226	0.069	-0.361	-0.091
Xptratio	-0.224	0.031	-0.285	-0.164
Xblack	0.092	0.027	0.040	0.145
Xlstat	-0.407	0.039	-0.485	-0.330

(c)

In this case with $n >> p$ the results of the Bayesian lasso are similar to those from the analysis with uninformative priors.

```
# Specify the model
model_string <- textConnection("model{
   # Likelihood
   for(i in 1:n){
     Y[i]   ~ dnorm(mu[i],tau)
     mu[i] <- alpha + inprod(X[i,],beta[])
   }
   for(j in 1:p){
      beta[j] ~ ddexp(0,taub)
   }
   alpha ~  dnorm(0, 0.01)
   tau   ~  dgamma(0.1, 0.1)
   taub  ~  dgamma(0.1, 0.1)
 }")

# Compile the model
model <- jags.model(model_string,data = data, n.chains=3,quiet=TRUE)

# Burn-in
update(model, 10000, progress.bar="none")

# Posterior draws
params  <- "beta"
```

```
samples <- coda.samples(model,
                        variable.names=params,
                        n.iter = 10000, progress.bar = "none")
```

```
post_sum3 <- summary(samples)
post_table3 <- cbind(post_sum3$statistics[,c("Mean", "SD")], post_sum3$quantiles[,c("2.5%", "97.5%")])
rownames(post_table3) <- colnames(X)
knitr::kable(post_table3, format = "markdown", digits = 3)
```

	Mean	SD	2.5%	97.5%
crim	-0.093	0.031	-0.153	-0.033
zn	0.105	0.035	0.036	0.172
indus	-0.001	0.042	-0.084	0.083
chas	0.075	0.024	0.027	0.122
nox	-0.205	0.048	-0.299	-0.109
rm	0.295	0.032	0.232	0.357
age	-0.002	0.037	-0.076	0.073
dis	-0.322	0.046	-0.412	-0.232
rad	0.242	0.064	0.113	0.368
tax	-0.182	0.069	-0.320	-0.044
ptratio	-0.219	0.031	-0.279	-0.158
black	0.090	0.027	0.037	0.142
lstat	-0.405	0.040	-0.483	-0.328