Problem 1

We compared Reggie Jackson's home run rate in the regular season and World Series. He hit 563 home runs in 2820 regular-season games and 10 home runs in 27 World Series games (a player can hit 0, 1, 2, ... home runs in a game). Assuming Uniform(0,10) priors for both home run rates, use JAGS to summarize the posterior distribution of (i) his home run rate in the regular season, (ii) his home run rate in the World Series, and (iii) the ratio of these rates. Provide trace plots for all three parameters and discuss convergence of the MCMC sampler including appropriate convergence diagnostics.

Solution:

We have Poisson likelihood and uniform prior:

```
library(rjags)
```

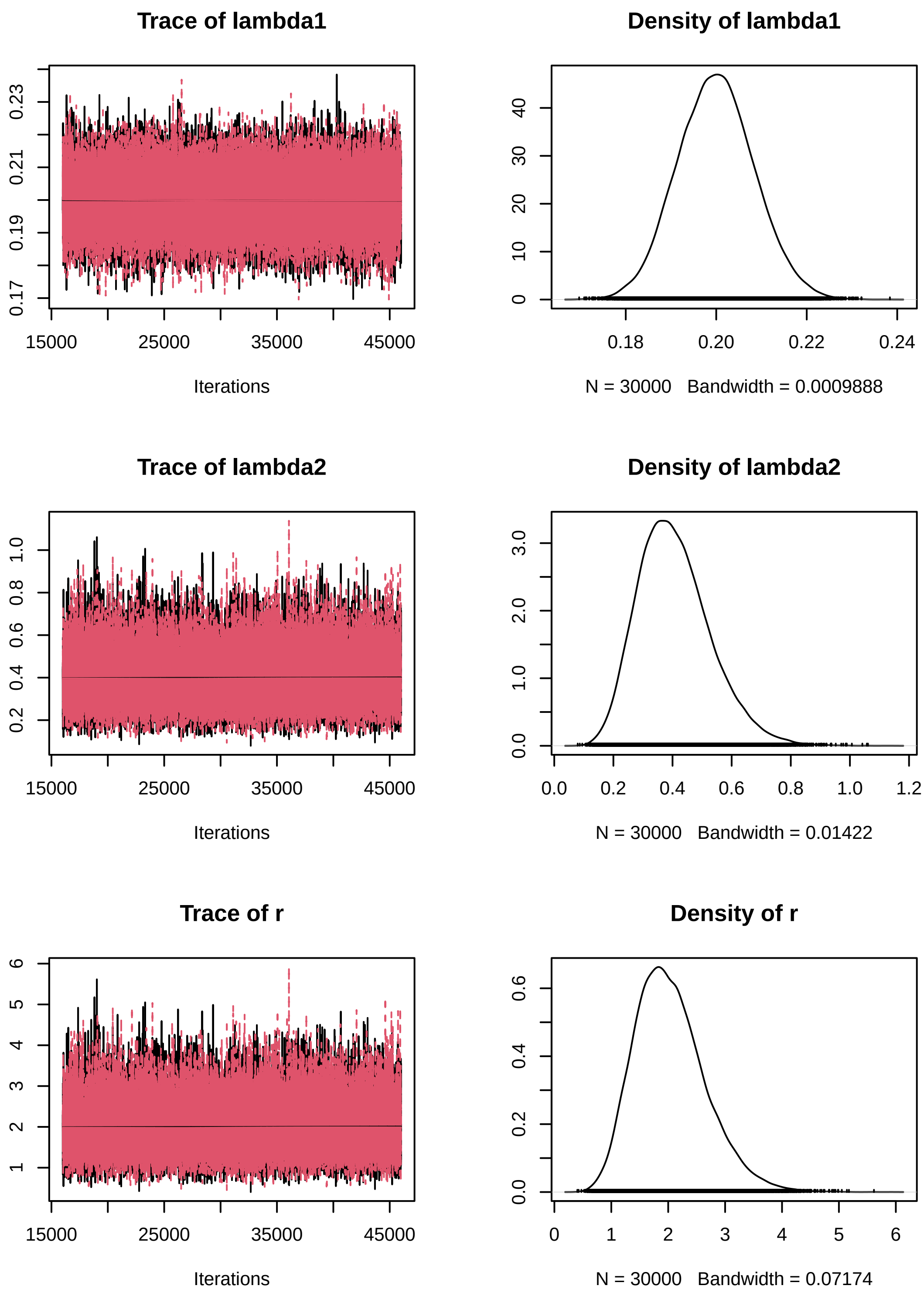
```
## Loading required package: coda

## Linked to JAGS 4.3.1

## Loaded modules: basemod,bugs
```

```
# Given parameters
N1 = 2820; Y1 = 563; N2 = 27; Y2 = 10
# Define string model
model_string = textConnection("model{
    # Likelihood
    Y1 ~ dpois(N1*lambda1)
    Y2 ~ dpois(N2*lambda2)
    # Priors
    lambda1 ~  dunif(0, 10)
    lambda2 ~  dunif(0, 10)
    r =lambda2/lambda1
 }")
# Initialize the parameters
inits      = list(lambda1= Y1/N1,lambda2 = Y2/N2)
# Load the data and compile the MCMC code
data       = list(N1 = N1,Y1 = Y1,N2 = N2,Y2 = Y2)
model      = jags.model(model_string,data = data, inits=inits, n.chains=2)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 2
##    Unobserved stochastic nodes: 2
##    Total graph size: 11
##
## Initializing model
```

```
#Burn-in for 15000 samples
update(model, 15000, progress.bar="none")
params     = c("lambda1","lambda2","r")
samples    = coda.samples(model,
             variable.names=params,
             n.iter=30000, progress.bar="none")
```

```r
plot(samples)
```




```r
summary(samples)
```

```
##
## Iterations = 16001:46000
## Thinning interval = 1
## Number of chains = 2
## Sample size per chain = 30000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##           Mean       SD  Naive SE Time-series SE
## lambda1 0.2000 0.008422 3.438e-05      4.383e-05
## lambda2 0.4064 0.122188 4.988e-04      6.851e-04
## r       2.0360 0.618846 2.526e-03      3.457e-03
##
## 2. Quantiles for each variable:
##
##           2.5%    25%    50%    75%  97.5%
## lambda1 0.1838 0.1942 0.1999 0.2056 0.2168
## lambda2 0.2025 0.3183 0.3944 0.4806 0.6787
## r       1.0144 1.5903 1.9738 2.4092 3.4191
```

```
effectiveSize(samples)
```

```
##  lambda1  lambda2        r
## 36948.23 31817.29 32066.32
```

```
gelman.diag(samples)
```

```
## Potential scale reduction factors:
##
##         Point est. Upper C.I.
## lambda1          1          1
## lambda2          1          1
## r                1          1
##
## Multivariate psrf
##
## 1
```

Since ESS is over 1000 and gelman gives a values less than 1.1 indicates convergence, we conclude we should get convergent trace which can also be seen from the plots.

Problem 2

A clinical trial gave six subjects a placebo and six subjects a new weight loss medication. The response variable is the change in weight (pounds) from baseline (so -2.0 means the subject lost 2 pounds). The data for the 12 subjects are:

Placebo	Treatment
2.0	-3.5
-3.1	-1.6
-1.0	-4.6

Placebo	Treatment
0.2	-0.9
0.3	-5.1
0.4	0.1

We conduct a Bayesian analysis to compare the means of these two groups. Would you say the treatment is effective? Is your conclusion sensitive to the prior?

Solution:

Assume two cases: i) two groups have the same variance and ii)two groups have the different variance.

In first case, let the placebo group is $Y_i \sim^{iid} \mathcal{N}(\mu, \sigma^2)$ for $i = 1, 2, ..., n_1$ and treatment group is $Y_i \sim^{iid} \mathcal{N}(\mu + \delta, \sigma^2)$ for $i = n_1 + 1, n_1 + 2, ..., n_1 + n_2 = n$.. The objective is to test whether $\delta = 0$ and thus the two groups have the same population mean. Since, the true variance of the groups are unknown we would like to use Jeffrey's prior $\pi(\mu, \delta, \sigma^2)$ the marginal posterior distribution of δ integrating over both μ and σ^2 is

$$\delta | rest \sim t_n \left[\bar{Y}_2 - \bar{Y}_1, \hat{\sigma}^2 \left(\frac{1}{n_1} + \frac{1}{n_2} \right) \right].$$

where $\bar{Y}_1$ and $\bar{Y}_2$ are the mean of Placebo and Treatment group, respectively.

In second case we assume $Y_i \sim^{iid} \mathcal{N}(\mu, \sigma_1^2)$ for $i = 1, 2, ..., n_1$ and $Y_i \sim^{iid} \mathcal{N}(\mu + \delta, \sigma_2^2)$ for $i = n_1 + 1, n_1 + 2, ..., n_1 + n_2 = n$. Then the posterior can be approximated MCMC.

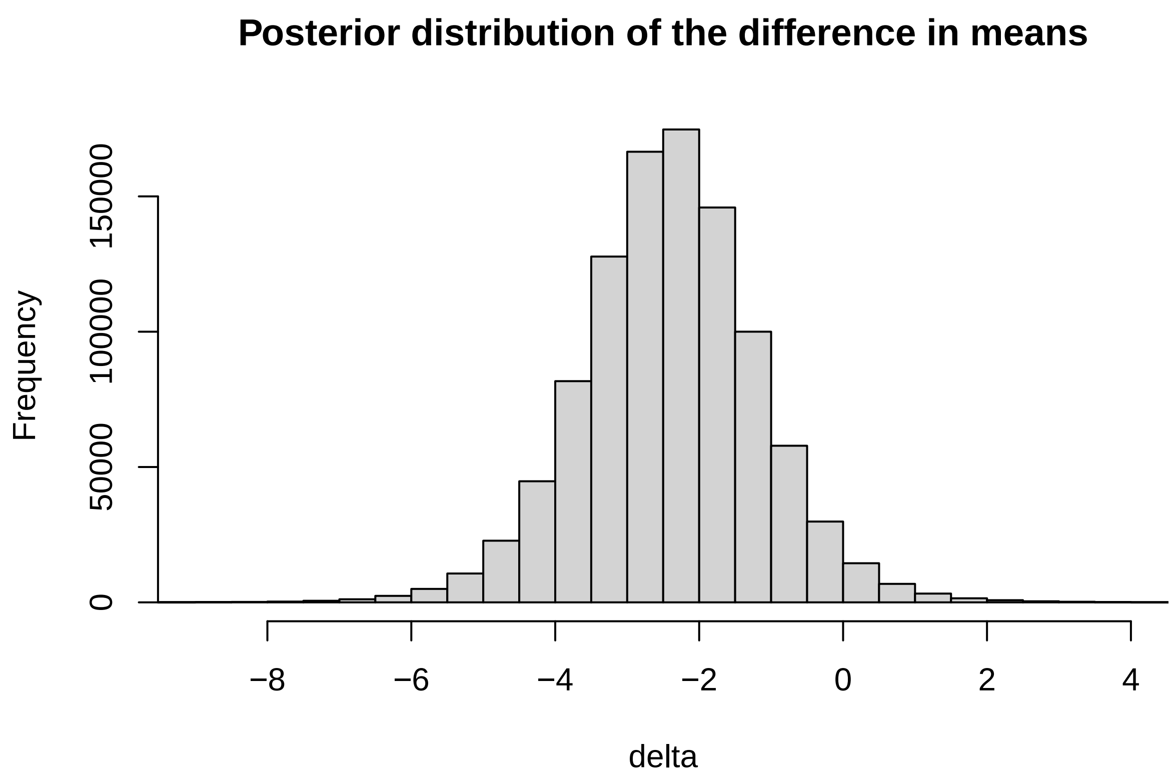
```
set.seed(100)
Y1 = c(2.0, -3.1, -1.0,0.2,0.3,0.4)
Y2 = c(-3.5, -1.6, -4.6,-0.9,-5.1,0.1)

Ybar1 = mean(Y1)
s21   = mean((Y1-Ybar1)^2)
n1    = length(Y1)


Ybar2 = mean(Y2)
s22   = mean((Y2-Ybar2)^2)
n2    =length(Y2)

# Posterior of the difference assuming equal variance
delta_hat =Ybar2-Ybar1
s2        =(n1*s21 + n2*s22)/(n1+n2)
scale     =sqrt(s2)*sqrt(1/n1+1/n2)
df        =n1+n2
cred_int  =delta_hat + scale*qt(c(0.025,0.975),df=df)
delta_hat
```

```
## [1] -2.4
```

```
cred_int
```

```
## [1] -4.6058799 -0.1941201
```

```r
# Posterior of delta assuming unequal variance using MC sampling
mu1       = Ybar1 + sqrt(s21/n1)*rt(1000000,df=n1)
mu2       = Ybar2 + sqrt(s22/n2)*rt(1000000,df=n2)
delta     = mu2-mu1

hist(delta,main="Posterior distribution of the difference in means",xlim = c(-9,4), breaks = 100)
```




```r
quantile(delta,c(0.025,0.975)) # 95% credible set
```

```
##        2.5%       97.5%
## -4.86590449  0.06850877
```

The credible set excludes zero and so there is some evidence that the mean differs by treatment group for the first case. For the second case, 0 is included in credible interval so the mean is not different for treatment group. In order to test sensitivity to the prior, we use vague but proper priors to fit the model.

```r
library(rjags)
data = list(n=6,Y1=Y1,Y2=Y2)

model_string = textConnection("model{

 # Likelihood
 for(i in 1:n){
   Y1[i] ~ dnorm(mu,tau)
```

```
   Y2[i] ~ dnorm(mu+delta,tau)
 }

 # Priors
 mu    ~  dnorm(0, 0.0001)
 delta ~  dnorm(0, 0.0001)
 tau   ~  dgamma(0.1, 0.1)
 sigma = 1/sqrt(tau)
}")

model   = jags.model(model_string,data = data, n.chains=2,quiet=TRUE)
update(model, 10000, progress.bar="none")
params  = c("delta")
samples = coda.samples(model,
          variable.names=params,
          n.iter=50000, progress.bar="none")
summary(samples)
```

```
##
## Iterations = 10001:60000
## Thinning interval = 1
## Number of chains = 2
## Sample size per chain = 50000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##           Mean             SD       Naive SE Time-series SE
##      -2.396197       1.228921       0.003886       0.006798
##
## 2. Quantiles for each variable:
##
##    2.5%     25%     50%     75%   97.5%
## -4.8512 -3.1681 -2.3960 -1.6348  0.0599
```

We observe that the results are slightly different as zero is included in the interval.

Problem 3

The response variable is `medv`, the median value of owner-occupied homes (in $1,000s), and the other 13 variables are covariates that describe the neighborhood.

(a) Fit a Bayesian linear regression model with uninformative Gaussian priors for the regression coefficients. Verify the MCMC sampler has converged, and summarize the posterior distribution of all regression coefficients.

Firstly, we analyze the data if some values are missed.

```
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

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

```
data(Boston)
summary(Boston)
```

```
##       crim                zn             indus            chas
##  Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   Min.   :0.00000
##  1st Qu.: 0.08205   1st Qu.:  0.00   1st Qu.: 5.19   1st Qu.:0.00000
##  Median : 0.25651   Median :  0.00   Median : 9.69   Median :0.00000
##  Mean   : 3.61352   Mean   : 11.36   Mean   :11.14   Mean   :0.06917
##  3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10   3rd Qu.:0.00000
##  Max.   :88.97620   Max.   :100.00   Max.   :27.74   Max.   :1.00000
##       nox               rm             age              dis
##  Min.   :0.3850   Min.   :3.561   Min.   :  2.90   Min.   : 1.130
##  1st Qu.:0.4490   1st Qu.:5.886   1st Qu.: 45.02   1st Qu.: 2.100
##  Median :0.5380   Median :6.208   Median : 77.50   Median : 3.207
##  Mean   :0.5547   Mean   :6.285   Mean   : 68.57   Mean   : 3.795
##  3rd Qu.:0.6240   3rd Qu.:6.623   3rd Qu.: 94.08   3rd Qu.: 5.188
##  Max.   :0.8710   Max.   :8.780   Max.   :100.00   Max.   :12.127
##       rad              tax           ptratio          black
##  Min.   : 1.000   Min.   :187.0   Min.   :12.60   Min.   :  0.32
##  1st Qu.: 4.000   1st Qu.:279.0   1st Qu.:17.40   1st Qu.:375.38
##  Median : 5.000   Median :330.0   Median :19.05   Median :391.44
##  Mean   : 9.549   Mean   :408.2   Mean   :18.46   Mean   :356.67
##  3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:396.23
##  Max.   :24.000   Max.   :711.0   Max.   :22.00   Max.   :396.90
##      lstat            medv
##  Min.   : 1.73   Min.   : 5.00
##  1st Qu.: 6.95   1st Qu.:17.02
##  Median :11.36   Median :21.20
##  Mean   :12.65   Mean   :22.53
##  3rd Qu.:16.95   3rd Qu.:25.00
##  Max.   :37.97   Max.   :50.00
```

So, it seems good. Next we construct Bayesian model with uninformative Gaussian prior.

```
Y = Boston%>%
  dplyr::select(medv)
Y = as.matrix(Y)
X = Boston%>%
  dplyr::select(-medv)
#Standardize covariates
X = scale(X)
X = cbind(1,X)
colnames(X)[1]="Intercept"
names=colnames(X)
#Load the data.
data = list(n=length(Y),p=ncol(X),Y=Y,X=X)
#define model string
model_string = textConnection("model{
# Likelihood
```

```
for(i in 1:n){
Y[i,] ~ dnorm(inprod(X[i,],beta[]),tau)
}
# Priors
beta[1]~dnorm(0, 0.0001)
for(j in 2:p){beta[j] ~ dnorm(0,taub*tau)}
tau ~ dgamma(0.1,0.1)
taub ~ dgamma(0.1, 0.1)
}")
model = jags.model(model_string, data = data, n.chains=2,quiet=TRUE)
update(model, 10000, progress.bar="none")
params = c("beta")
samples = coda.samples(model, variable.names=params, n.iter=10000,progress.bar="none")

effectiveSize(samples)
```

```
##    beta[1]    beta[2]    beta[3]    beta[4]    beta[5]    beta[6]    beta[7]    beta[8]
## 19764.516  7248.796  5234.238  3170.714 15211.759  3859.738  5411.376  4322.829
##    beta[9]   beta[10]   beta[11]   beta[12]   beta[13]   beta[14]
##  4015.823  1529.352  1316.833  6785.301 13018.792  4838.178
```

```
gelman.diag(samples)
```

```
## Potential scale reduction factors:
##
##          Point est. Upper C.I.
## beta[1]           1       1.00
## beta[2]           1       1.00
## beta[3]           1       1.00
## beta[4]           1       1.00
## beta[5]           1       1.00
## beta[6]           1       1.00
## beta[7]           1       1.00
## beta[8]           1       1.00
## beta[9]           1       1.00
## beta[10]          1       1.01
## beta[11]          1       1.01
## beta[12]          1       1.00
## beta[13]          1       1.00
## beta[14]          1       1.00
##
## Multivariate psrf
##
## 1
```

```
sum                         = summary(samples)
rownames(sum$statistics) = names
rownames(sum$quantiles)  = names
sum$statistics              = round(sum$statistics,4)
sum$quantiles               = round(sum$quantiles,4)
sum
```

```
##
## Iterations = 10001:20000
## Thinning interval = 1
## Number of chains = 2
## Sample size per chain = 10000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##              Mean     SD Naive SE Time-series SE
## Intercept 22.5331 0.2122   0.0015         0.0015
## crim      -0.8881 0.2775   0.0020         0.0033
## zn         1.0154 0.3172   0.0022         0.0044
## indus      0.0338 0.4078   0.0029         0.0072
## chas       0.6974 0.2157   0.0015         0.0018
## nox       -1.9165 0.4271   0.0030         0.0069
## rm         2.7161 0.2889   0.0020         0.0039
## age       -0.0044 0.3625   0.0026         0.0055
## dis       -2.9655 0.4093   0.0029         0.0065
## rad        2.3253 0.5551   0.0039         0.0142
## tax       -1.7701 0.6000   0.0042         0.0165
## ptratio   -2.0185 0.2797   0.0020         0.0034
## black      0.8491 0.2429   0.0017         0.0021
## lstat     -3.6834 0.3568   0.0025         0.0051
##
## 2. Quantiles for each variable:
##
##              2.5%     25%     50%     75%   97.5%
## Intercept 22.1205 22.3883 22.5318 22.6759 22.9512
## crim      -1.4296 -1.0765 -0.8877 -0.7002 -0.3439
## zn         0.3949  0.8049  1.0116  1.2295  1.6490
## indus     -0.7785 -0.2393  0.0366  0.3099  0.8193
## chas       0.2706  0.5524  0.6981  0.8447  1.1175
## nox       -2.7631 -2.1991 -1.9131 -1.6294 -1.0875
## rm         2.1462  2.5238  2.7147  2.9094  3.2811
## age       -0.7151 -0.2496 -0.0054  0.2425  0.7008
## dis       -3.7549 -3.2442 -2.9665 -2.6891 -2.1652
## rad        1.2464  1.9481  2.3251  2.6987  3.4132
## tax       -2.9518 -2.1686 -1.7701 -1.3685 -0.5952
## ptratio   -2.5706 -2.2048 -2.0170 -1.8285 -1.4724
## black      0.3700  0.6868  0.8499  1.0114  1.3259
## lstat     -4.3906 -3.9225 -3.6835 -3.4443 -2.9830
```

Since the effective size is more than 1000 and and the Gelman-Rubin statistics are 1.0, the chains have clearly converged. Also, credible intervals include zero for the predictors indus and age so they are not significant.

(b) Perform a classic least squares analysis (e.g., using the lm function in R). Compare the results numerically and conceptually with the Bayesian results.

```
ols_data = cbind(Y,X[,2:14])
ols_data = data.frame(ols_data)
ols_model = lm(medv~.-medv, data = ols_data)
#ols_model$coefficients
tidy(ols_model)
```

```
## # A tibble: 14 x 5
##    term        estimate std.error statistic  p.value
##    <chr>          <dbl>     <dbl>     <dbl>    <dbl>
##  1 (Intercept)  22.5        0.211   107.     0
##  2 crim         -0.929      0.283    -3.29   1.09e- 3
##  3 zn            1.08       0.320     3.38   7.78e- 4
##  4 indus         0.141      0.422     0.334  7.38e- 1
##  5 chas          0.682      0.219     3.12   1.93e- 3
##  6 nox          -2.06       0.443    -4.65   4.25e- 6
##  7 rm            2.68       0.294     9.12   1.98e-18
##  8 age           0.0195     0.372     0.0524 9.58e- 1
##  9 dis          -3.11       0.420    -7.40   6.01e-13
## 10 rad           2.66       0.578     4.61   5.07e- 6
## 11 tax          -2.08       0.634    -3.28   1.11e- 3
## 12 ptratio      -2.06       0.283    -7.28   1.31e-12
## 13 black         0.850      0.245     3.47   5.73e- 4
## 14 lstat        -3.75       0.362   -10.3    7.78e-23
```

They are numerically almost equivalent. However, in Bayesian approach this is parameter rather than fixed number. Hence, all coefficients have distributions. Even though, their representation are same, their interpretations are completely different. We again see that indus and age are not significant since their intervals have zero.

(c) Refit the Bayesian model with double exponential priors for the regression coefficients, and discuss how the results differ from the analysis with uninformative priors.

```
model_string = textConnection("model{
 # Likelihood
  for(i in 1:n){
    Y[i,] ~ dnorm(inprod(X[i,],beta[]),taue)
  }
 # Priors
 beta[1] ~ dnorm(0,0.001)
  for(j in 2:p){
    beta[j] ~ ddexp(0,taue*taub)
  }
  taue  ~ dgamma(0.1, 0.1)
  taub  ~ dgamma(0.1, 0.1)
}")

model = jags.model(model_string,data = data, n.chains = 2,quiet=TRUE)
params  = c("beta")
update(model, 10000, progress.bar="none")
samples2 = coda.samples(model, variable.names=params, n.iter=20000,progress.bar="none")
```

```
effectiveSize(samples2)
```

```
##   beta[1]   beta[2]   beta[3]   beta[4]   beta[5]   beta[6]   beta[7]   beta[8]
## 40000.000  9883.721  6716.648  4579.054 20945.183  4275.957  7148.601  6476.765
##   beta[9]  beta[10]  beta[11]  beta[12]  beta[13]  beta[14]
##  4534.781  1870.881  1792.615  8621.853 15931.105  6424.295
```

```r
gelman.diag(samples2)
```

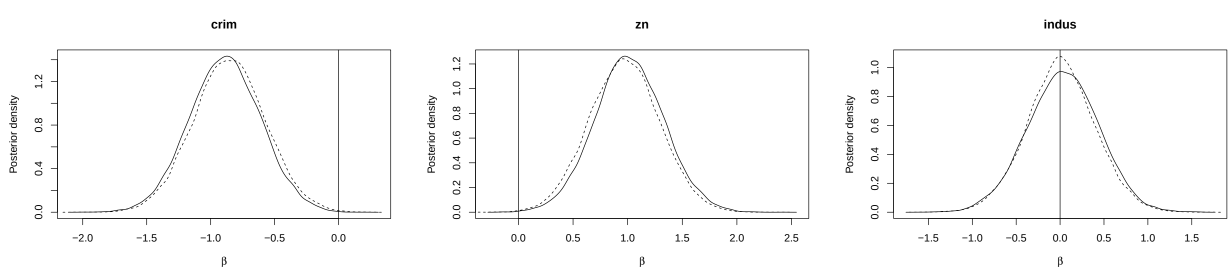
```
## Potential scale reduction factors:
##
##          Point est. Upper C.I.
## beta[1]           1          1
## beta[2]           1          1
## beta[3]           1          1
## beta[4]           1          1
## beta[5]           1          1
## beta[6]           1          1
## beta[7]           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
## beta[14]          1          1
##
## Multivariate psrf
##
## 1
```

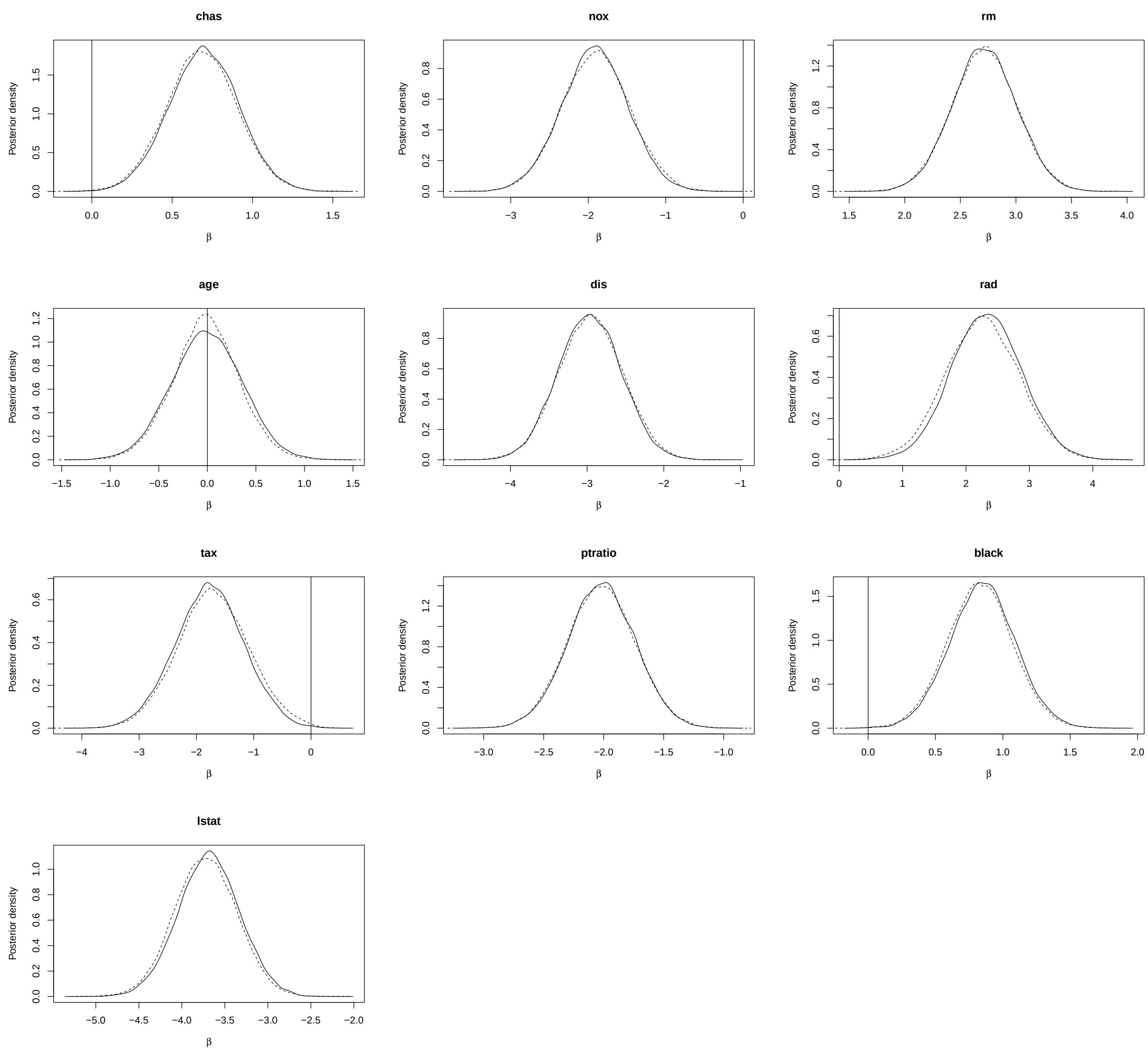
```r
sum                        = summary(samples2)
rownames(sum$statistics) = names
rownames(sum$quantiles)  = names
sum$statistics             = round(sum$statistics,4)
sum$quantiles              = round(sum$quantiles,4)
sum
```

```
##
## Iterations = 11001:31000
## Thinning interval = 1
## Number of chains = 2
## Sample size per chain = 20000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##              Mean     SD Naive SE Time-series SE
## Intercept 22.5314 0.2119   0.0011         0.0011
## crim      -0.8537 0.2824   0.0014         0.0028
## zn         0.9694 0.3215   0.0016         0.0039
## indus      0.0114 0.3913   0.0020         0.0058
## chas       0.6847 0.2195   0.0011         0.0015
## nox       -1.9015 0.4385   0.0022         0.0067
## rm         2.7148 0.2925   0.0015         0.0035
## age       -0.0147 0.3398   0.0017         0.0042
## dis       -2.9539 0.4185   0.0021         0.0062
## rad        2.2553 0.5809   0.0029         0.0134
## tax       -1.7016 0.6273   0.0031         0.0148
## ptratio   -2.0212 0.2844   0.0014         0.0031
```

```
## black       0.8299 0.2428   0.0012         0.0019
## lstat      -3.7231 0.3604   0.0018         0.0045
##
## 2. Quantiles for each variable:
##
##              2.5%     25%     50%     75%   97.5%
## Intercept 22.1177 22.3886 22.5309 22.6742 22.9481
## crim      -1.4064 -1.0433 -0.8546 -0.6649 -0.2967
## zn         0.3411  0.7517  0.9703  1.1844  1.6013
## indus     -0.7648 -0.2456  0.0088  0.2672  0.7908
## chas       0.2544  0.5380  0.6843  0.8320  1.1154
## nox       -2.7582 -2.2004 -1.9002 -1.6071 -1.0363
## rm         2.1386  2.5181  2.7163  2.9120  3.2894
## age       -0.6898 -0.2363 -0.0140  0.2082  0.6579
## dis       -3.7704 -3.2357 -2.9534 -2.6698 -2.1389
## rad        1.1090  1.8623  2.2567  2.6498  3.3941
## tax       -2.9248 -2.1200 -1.7120 -1.2807 -0.4466
## ptratio   -2.5766 -2.2140 -2.0212 -1.8300 -1.4629
## black      0.3526  0.6670  0.8297  0.9909  1.3111
## lstat     -4.4268 -3.9671 -3.7233 -3.4772 -3.0208
```

```r
for(j in 2:14){
# Collect the MCMC iteration from both chains for the two priors
s1 = c(samples[[1]][,j],samples[[2]][,j])
s2 = c(samples2[[1]][,j],samples2[[2]][,j])

# Get smooth density estimate for each prior
d1 = density(s1)
d2 = density(s2)
# Plot the density estimates
mx = max(c(d1$y,d2$y))
plot(d1$x,d1$y,type="l",ylim=c(0,mx),xlab=expression(beta),ylab="Posterior density",main=names[j])
lines(d2$x,d2$y,lty=2)
abline(v=0)
legend(1, 95, legend=c("Uninformative Gaussian", "Bayesian LASSO"),
     col=c("red", "blue"), lty=1:2, cex=0.8)
}
```





Since we have enough data points the choice of prior have only minor affect. It also shown in figures.

(d) Fit a Bayesian linear regression model in (a) using only the first 500 observations and compute the posterior predictive distribution for the final 6 observations. Plot the posterior predictive distribution versus the actual value for these 6 observations and comment on whether the predictions are reasonable.

```
Y_train = Y[1:500,]
Y_test  = Y[501:506,]
X_train = X[1:500,]
X_test  = X[501:506,]
n_train = length(Y_train)
n_test  = length(Y_test)
p       = ncol(X_train)

model_string <- textConnection("model{
   # Likelihood
  for(i in 1:no){
```

```
    Yo[i]   ~ dnorm(muo[i],inv.var)
    muo[i] = inprod(Xo[i,],beta[])
  }

  # Prediction
  for(i in 1:np){
    Y_test[i]  ~ dnorm(mup[i],inv.var)
    mup[i] = inprod(Xp[i,],beta[])
  }

  # Priors
beta[1] ~ dnorm(0,0.001)
for(j in 2:p){beta[j] ~ dnorm(0,taub*inv.var)}
taub ~ dgamma(0.1, 0.1)
inv.var   ~ dgamma(0.01, 0.01)
sigma     = 1/sqrt(inv.var)
}")

data  = list(Yo=Y_train,no=n_train,np=n_test,p=p,Xo=X_train,Xp=X_test)
model = jags.model(model_string, data = data)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 500
##    Unobserved stochastic nodes: 22
##    Total graph size: 8630
##
## Initializing model
```

```
update(model, 10000, progress.bar="none")
samp = coda.samples(model,
        variable.names=c("beta","sigma","Y_test"),
        n.iter=20000, progress.bar="none")

summary(samp[,-c(1:n_test)])
```

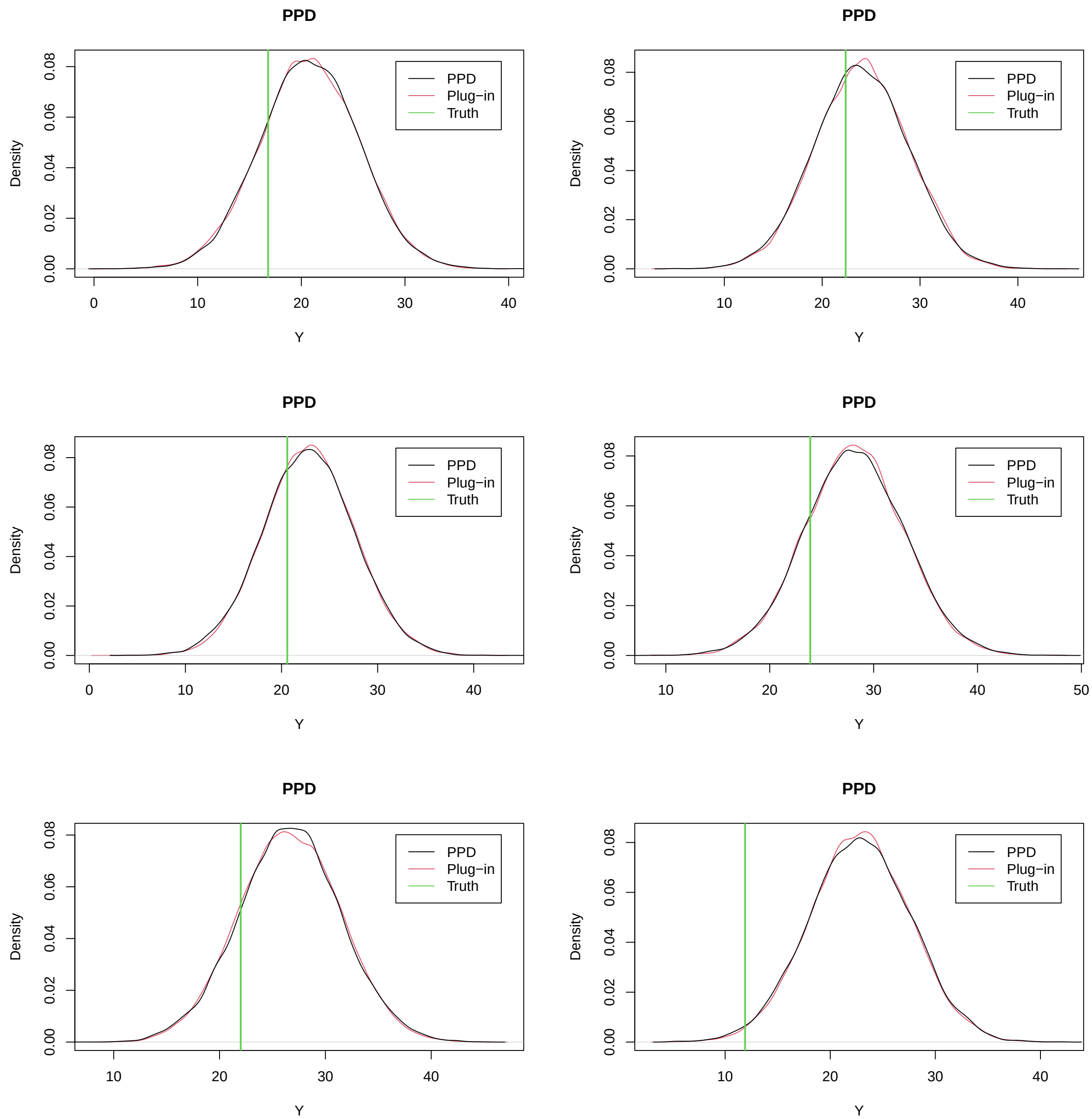
```
##
## Iterations = 10001:30000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 20000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##               Mean     SD Naive SE Time-series SE
## beta[1]  22.58579 0.2141 0.001514       0.001514
## beta[2]  -0.86881 0.2792 0.001974       0.003083
## beta[3]   1.05588 0.3125 0.002210       0.004272
## beta[4]   0.01935 0.4140 0.002927       0.006905
```

```
## beta[5]   0.68016 0.2164 0.001530       0.001764
## beta[6]  -1.84645 0.4249 0.003004       0.007333
## beta[7]   2.71645 0.2862 0.002024       0.003766
## beta[8]   0.02889 0.3632 0.002569       0.005683
## beta[9]  -3.04706 0.4103 0.002901       0.006493
## beta[10]  2.19156 0.5619 0.003973       0.014332
## beta[11] -1.80001 0.6068 0.004291       0.015363
## beta[12] -1.88065 0.2849 0.002015       0.003527
## beta[13]  0.84576 0.2439 0.001725       0.002248
## beta[14] -3.76986 0.3585 0.002535       0.005259
## sigma     4.74280 0.1507 0.001066       0.001144
##
## 2. Quantiles for each variable:
##
##             2.5%     25%      50%     75%   97.5%
## beta[1]  22.1671 22.4419 22.58564 22.7299 23.0044
## beta[2]  -1.4198 -1.0550 -0.86847 -0.6787 -0.3250
## beta[3]   0.4356  0.8463  1.05693  1.2707  1.6614
## beta[4]  -0.7915 -0.2582  0.01808  0.3006  0.8292
## beta[5]   0.2554  0.5324  0.68000  0.8278  1.1044
## beta[6]  -2.6689 -2.1357 -1.84501 -1.5603 -0.9991
## beta[7]   2.1538  2.5226  2.71809  2.9091  3.2747
## beta[8]  -0.6866 -0.2143  0.02653  0.2760  0.7404
## beta[9]  -3.8588 -3.3236 -3.04861 -2.7671 -2.2462
## beta[10]  1.1206  1.8091  2.18737  2.5650  3.3191
## beta[11] -2.9732 -2.2094 -1.80758 -1.3921 -0.6174
## beta[12] -2.4344 -2.0730 -1.88101 -1.6883 -1.3213
## beta[13]  0.3721  0.6813  0.84512  1.0114  1.3264
## beta[14] -4.4718 -4.0136 -3.77104 -3.5242 -3.0690
## sigma     4.4568  4.6392  4.73969  4.8416  5.0504
```

```r
samps           = samp[[1]]
Y_test.samps    = samps[,1:n_test]
beta.samps      = samps[,n_test+1:p]
sigma.samps     = samps[,ncol(samps)]

# Compute the posterior mean for the plug-in predictions

beta.mn         = colMeans(beta.samps)
sigma.mn        = mean(sigma.samps)
```

```r
# Plot the PPD and plug-in
 for(j in 1:6){
    # Plug-in
    mu <- sum(X_test[j,]*beta.mn)
    y  <- rnorm(20000,mu,sigma.mn)
    plot(density(y),col=2,xlab="Y",main="PPD")

    # PPD
    lines(density(Y_test.samps[,j]))

    # Truth
    abline(v=Y_test[j],col=3,lwd=2)
    legend("topright",c("PPD","Plug-in","Truth"),col=1:3,lty=1,inset=0.05)
```

```
}
```




```
# plug-in 95% intervals
low1   =  X_test%*%beta.mn - 1.96*sigma.mn
high1  =  X_test%*%beta.mn + 1.96*sigma.mn
cover1 =  mean(Y_test>low1 & Y_test<high1)
mean(cover1)
```

```
## [1] 0.8333333
```

```r
# PPD 95% intervals
low2   =  apply(Y_test.samps,2,quantile,0.025)
high2  =  apply(Y_test.samps,2,quantile,0.975)
cover2 =  mean(Y_test>low2 & Y_test<high2)
mean(cover2)
```

```
## [1] 0.8333333
```

From plots we observe that both plug-in prediction and PPD give reasonable predictions.