Mth422a- Assignment-4

Vijay Soren (211163)

2024-03-28

Que 5

part (a)

Uninformative Gaussian prior

β_0 follow $N(0, 100^2)$

β_j follow $N(0, 100^2)$ for j = 1,2,..p

```
library(MASS)
library(rjags)
```

```
## Warning: package 'rjags' was built under R version 4.3.3

## Loading required package: coda

## Warning: package 'coda' was built under R version 4.3.3

## Linked to JAGS 4.3.1

## Loaded modules: basemod,bugs
```

```
data("Boston")
Y <- Boston$medv
X <- Boston[,1:13]
X <- as.matrix(X)
X <- scale(X)
Intercept <- rep(1,length(Y))
X <- cbind(Intercept,X)

data <- list(Y = Y, X = X, n = length(Y), p = dim(X)[2])
model_string <- textConnection("model{
    # liklihood
    for(i in 1:n){
    Y[i] ~ dnorm(inprod(X[i,],beta[]), tau)
    }

    #priors
    for(j in 1:p){
```

```
    beta[j] ~ dnorm(0,0.0001)
    }
    tau ~ dgamma(0.01,0.01)
}")


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=20000, progress.bar="none")
Beyasian_betas <- summary(samples)$statistics
Beyasian_betas <- Beyasian_betas[,1]
Beyasian_betas <- as.numeric(Beyasian_betas)
Beyasian_betas
```

```
##  [1] 22.5325995 -0.9296170  1.0802141  0.1451707  0.6818630 -2.0636852
##  [7]  2.6776606  0.0217636 -3.1016132  2.6706139 -2.0835712 -2.0648820
## [13]  0.8484025 -3.7447547
```

part (b)

```
model1 <- lm(Y ~ X)
summary(model1)
```

```
##
## Call:
## lm(formula = Y ~ X)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.595  -2.730  -0.518   1.777  26.199
##
## Coefficients: (1 not defined because of singularities)
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 22.53281    0.21095 106.814  < 2e-16 ***
## XIntercept        NA         NA      NA       NA
## Xcrim       -0.92906    0.28269  -3.287 0.001087 **
## Xzn          1.08264    0.32016   3.382 0.000778 ***
## Xindus       0.14104    0.42188   0.334 0.738288
## Xchas        0.68241    0.21884   3.118 0.001925 **
## Xnox        -2.05875    0.44262  -4.651 4.25e-06 ***
## Xrm          2.67688    0.29364   9.116  < 2e-16 ***
## Xage         0.01949    0.37184   0.052 0.958229
## Xdis        -3.10712    0.41999  -7.398 6.01e-13 ***
## Xrad         2.66485    0.57770   4.613 5.07e-06 ***
## Xtax        -2.07884    0.63379  -3.280 0.001112 **
## Xptratio    -2.06265    0.28323  -7.283 1.31e-12 ***
## Xblack       0.85011    0.24521   3.467 0.000573 ***
## Xlstat      -3.74733    0.36216 -10.347  < 2e-16 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.745 on 492 degrees of freedom
## Multiple R-squared:  0.7406, Adjusted R-squared:  0.7338
## F-statistic: 108.1 on 13 and 492 DF,  p-value: < 2.2e-16
```

```
classic_betas <- model1$coefficients
classic_betas <- classic_betas[-2]
classic_betas <- as.numeric(classic_betas)
### comapre results
coff <- cbind(Beyasian_betas,classic_betas)
print(coff)
```

```
##       Beyasian_betas classic_betas
##  [1,]     22.5325995   22.53280632
##  [2,]     -0.9296170   -0.92906457
##  [3,]      1.0802141    1.08263896
##  [4,]      0.1451707    0.14103943
##  [5,]      0.6818630    0.68241438
##  [6,]     -2.0636852   -2.05875361
##  [7,]      2.6776606    2.67687661
##  [8,]      0.0217636    0.01948534
##  [9,]     -3.1016132   -3.10711605
## [10,]      2.6706139    2.66485220
## [11,]     -2.0835712   -2.07883689
## [12,]     -2.0648820   -2.06264585
## [13,]      0.8484025    0.85010886
## [14,]     -3.7447547   -3.74733185
```

part (c)

```
Y <- Boston$medv
X <- Boston[,1:13]
X <- as.matrix(X)
X <- scale(X)
Intercept <- rep(1,length(Y))
X <- cbind(Intercept,X)

data <- list(Y = Y, X = X, n = length(Y), p = dim(X)[2])
model_string <- textConnection("model{
    # liklihood
    for(i in 1:n){
    Y[i] ~ dnorm(inprod(X[i,],beta[]), tau)
    }

    #priors
    for(j in 1:p){
    beta[j] ~ ddexp(0,0.0001)
    }
    tau ~ dgamma(0.01,0.01)
```

```r
}")


model <- jags.model(model_string,data = data, n.chains=1,quiet=TRUE)
update(model, 10000, progress.bar="none")
params  <- c("beta")
samples <- coda.samples(model,
                        variable.names=params,
                        n.iter=20000, progress.bar="none")
B_betas_with_ddexp <- summary(samples)$statistics
B_betas_eith_ddexp <- B_betas_with_ddexp[,1]
B_betas_with_ddexp <- as.numeric(B_betas_with_ddexp)
B_betas_with_ddexp <- B_betas_with_ddexp[1:14]
coff1 <- cbind(Beyasian_betas,B_betas_with_ddexp)
print(coff1)
```

```
##       Beyasian_betas B_betas_with_ddexp
##  [1,]     22.5325995        22.53101516
##  [2,]     -0.9296170        -0.93420739
##  [3,]      1.0802141         1.07632305
##  [4,]      0.1451707         0.13873416
##  [5,]      0.6818630         0.68327174
##  [6,]     -2.0636852        -2.07100332
##  [7,]      2.6776606         2.67398471
##  [8,]      0.0217636         0.01809408
##  [9,]     -3.1016132        -3.11247444
## [10,]      2.6706139         2.66209872
## [11,]     -2.0835712        -2.06161448
## [12,]     -2.0648820        -2.06981242
## [13,]      0.8484025         0.85145890
## [14,]     -3.7447547        -3.74813594
```

part (d)

```r
#### part d
library(MASS)
library(rjags)
data("Boston")

Y <- Boston$medv
Y <- Y[1:500]
X <- Boston[1:500,1:13]
X <- as.matrix(X)
X <- scale(X)
X_pred <- Boston[501:506,1:13]
X_pred <- as.matrix(X_pred)
X_pred <- scale(X_pred)
### nan
X_pred[,2] <- 0
X_pred[,4] <- 0
n_pred <- 6
```

```r
Intercept <- rep(1,length(Y))
X <- cbind(Intercept,X)
X_pred <- cbind(Intercept,X_pred)
```

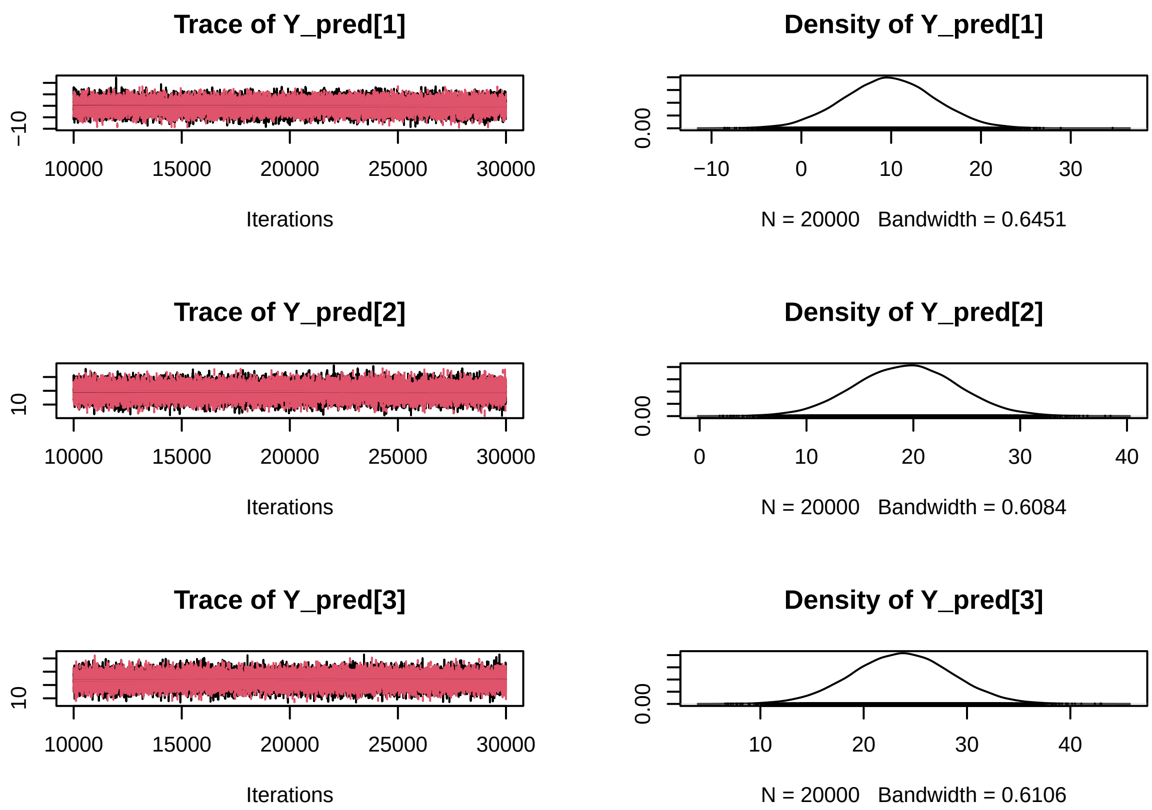
```
## Warning in cbind(Intercept, X_pred): number of rows of result is not a multiple
## of vector length (arg 1)
```

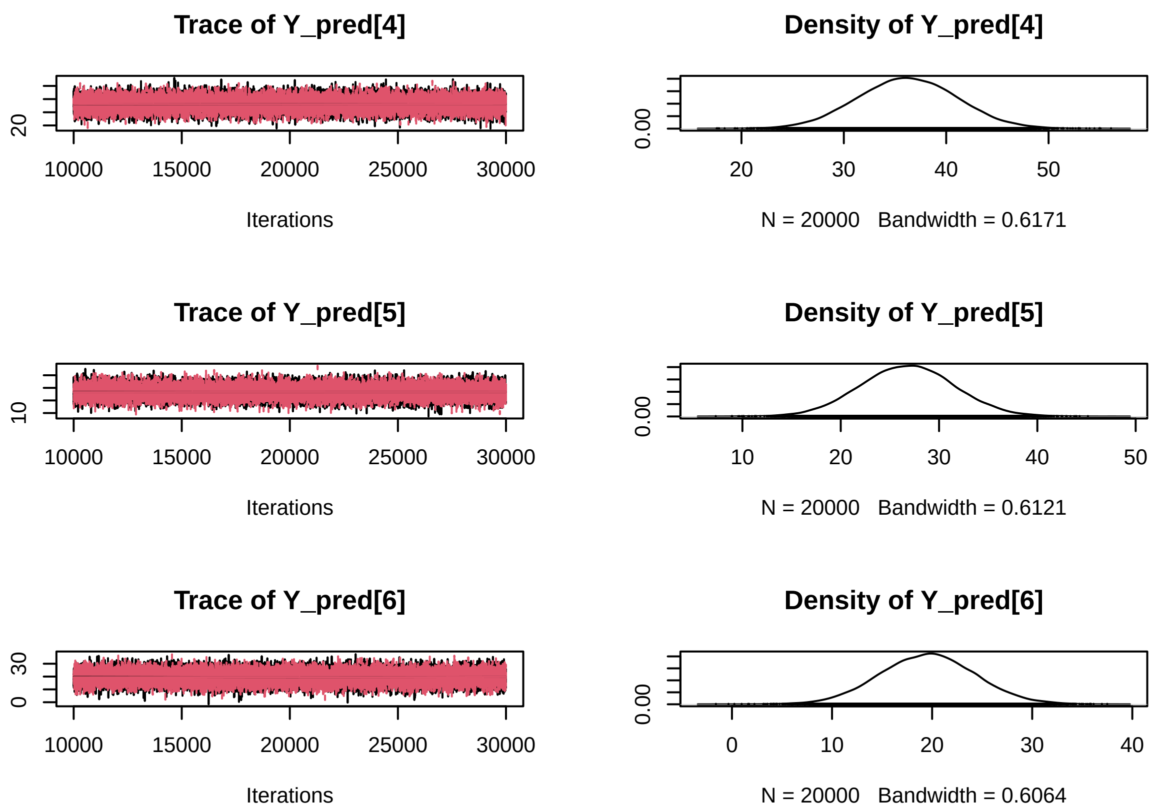
```r
data <- list(Y = Y, X = X, n = length(Y), p = dim(X)[2], X_pred = X_pred, n_pred = n_pred)
model_string <- textConnection("model{
    # liklihood
    for(i in 1:n){
    Y[i] ~ dnorm(inprod(X[i,],beta[]), tau)
    }

    #priors
    beta[1]~dnorm(0,0.0001)
    for(j in 2:p){
    beta[j] ~ dnorm(0,0.0001)
    }
    tau ~ dgamma(0.01,0.01)

    #predictions
    for(i in 1:n_pred){
    Y_pred[i] ~ dnorm(inprod(X_pred[i,],beta[]), tau)
    }
}")


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

```
##
## Iterations = 10001:30000
## 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
## Y_pred[1]  9.77 5.080  0.02540        0.03055
## Y_pred[2] 19.36 4.779  0.02389        0.02536
## Y_pred[3] 23.71 4.796  0.02398        0.02412
## Y_pred[4] 36.23 4.847  0.02423        0.02524
## Y_pred[5] 26.81 4.808  0.02404        0.02471
## Y_pred[6] 19.56 4.762  0.02381        0.02418
```

```
##
## 2. Quantiles for each variable:
##
##                2.5%   25%   50%   75% 97.5%
## Y_pred[1] -0.1648  6.39  9.78 13.18 19.68
## Y_pred[2] 10.0320 16.11 19.38 22.58 28.77
## Y_pred[3] 14.2499 20.46 23.71 26.92 33.07
## Y_pred[4] 26.7619 32.94 36.22 39.51 45.76
## Y_pred[5] 17.4345 23.58 26.82 30.02 36.24
## Y_pred[6] 10.2542 16.34 19.58 22.73 28.95
```

```
plot(samples)
```





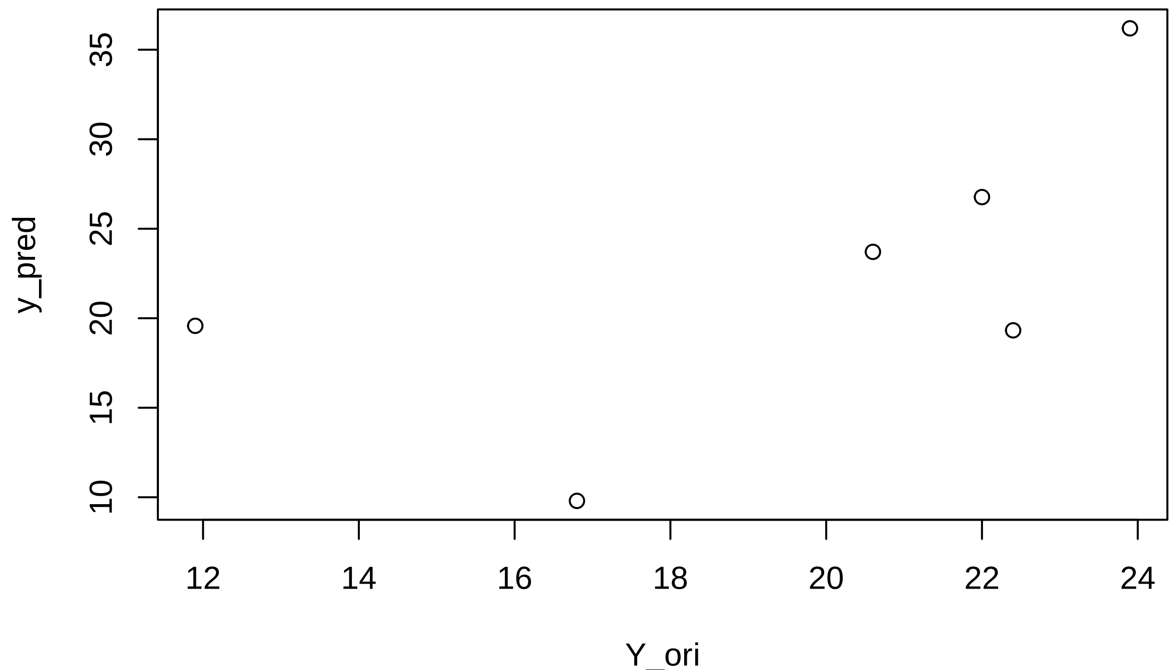

```
### we get Y_pred values from above model
y_pred <- c(9.798,19.324,23.708,36.195,26.768,19.578)
y_pred
```

```
## [1]  9.798 19.324 23.708 36.195 26.768 19.578
```

```
Y_ori <- Boston[501:506,14]
Y_ori
```

```
## [1] 16.8 22.4 20.6 23.9 22.0 11.9
```

```
plot(Y_ori,y_pred)
```



y_pred
10
15
20
25
30
35
12
14
16
18
20
22
24
Y_ori