# 682hw5

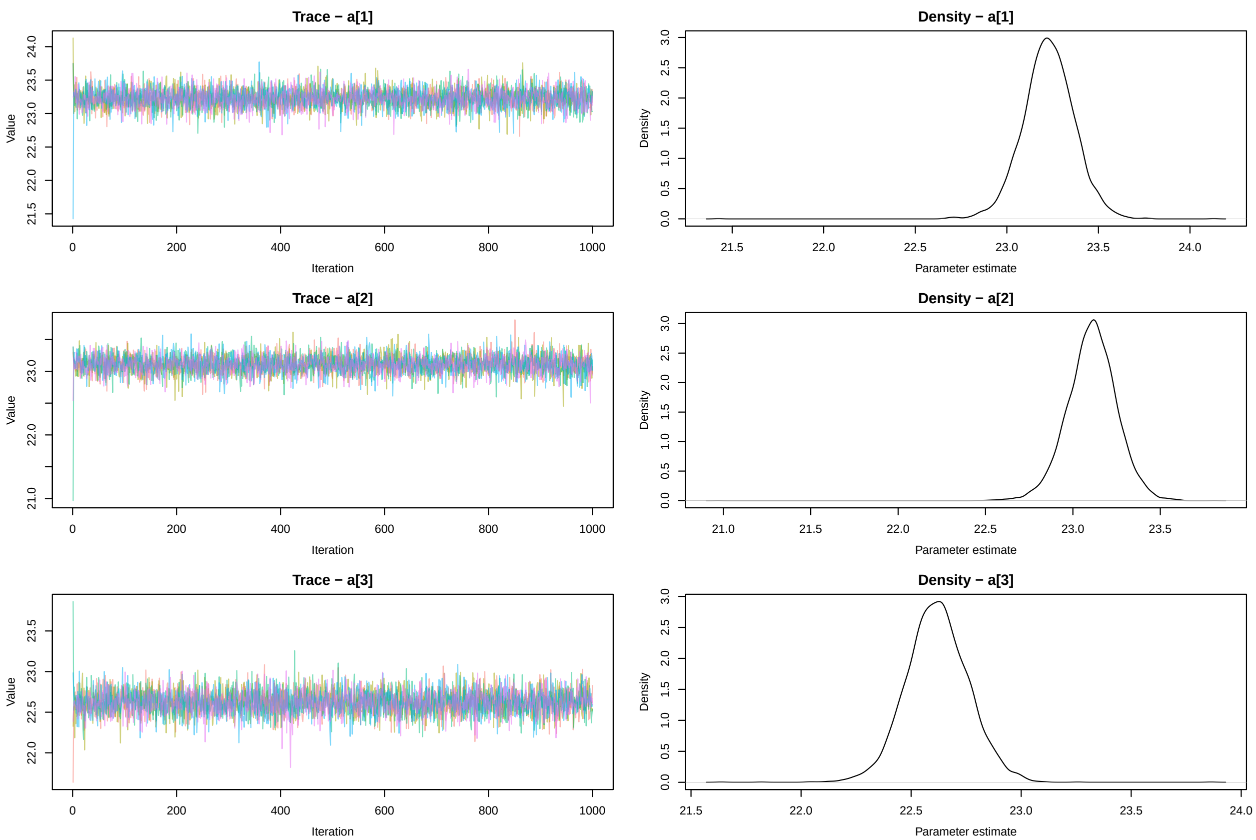
*Zheng Xu*

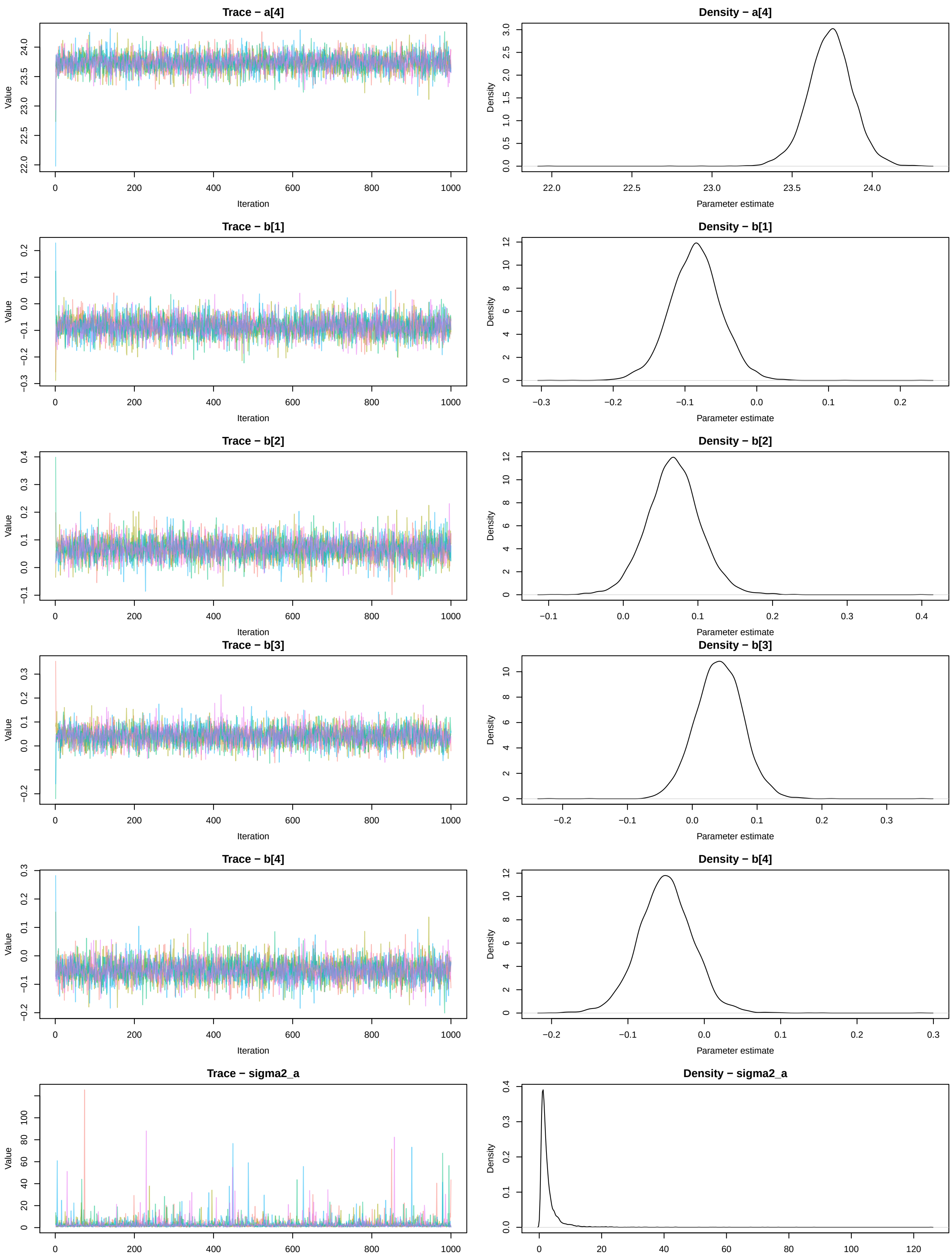
*December 2, 2017*

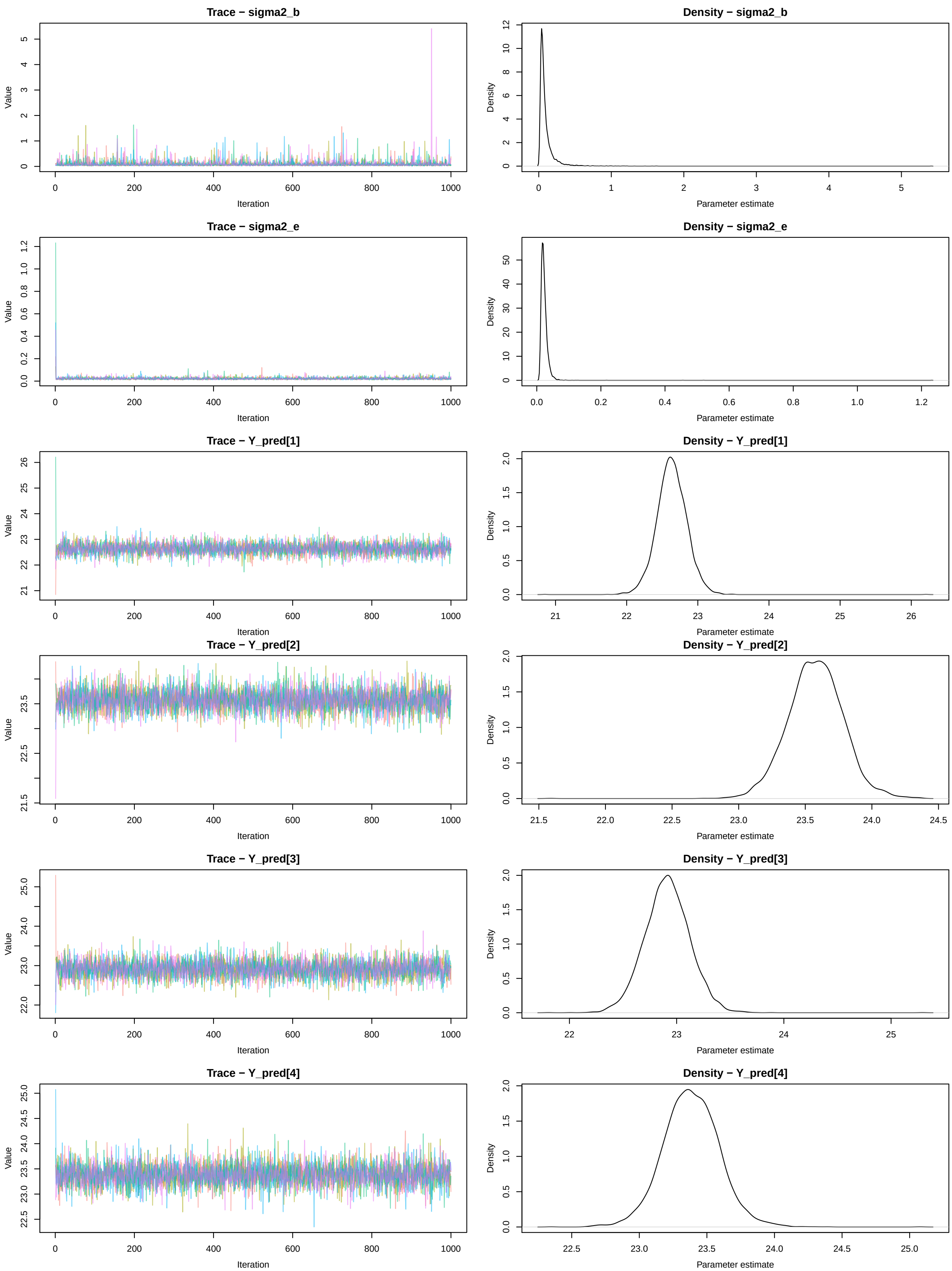
```r
knitr::opts_chunk$set(fig.width=12, fig.height=8, warning=FALSE, message=FALSE)
library(R2jags)
library(MASS)
library(ggplot2)
library(mcmcplots)
library(MCMCvis)
library(geoR)

load("swim_time.RData")
ns   <- nrow(Y)
nt   <- ncol(Y)
JAGS_swimmer_model = function(){
  for (i in 1:ns) {
    for (j in 1:nt) {
      Y[i, j]    ~ dnorm(mean[i, j], tau_e)
      mean[i, j] <- a[i] + b[i] *j
    }

  }
  for (i in 1:ns){
    a[i]~ dnorm(22, tau_a)
    b[i]~ dnorm(0, tau_b)
    Y_pred[i] ~ dnorm(a[i] + b[i] * 7, tau_e)
    z[i] <- (Y_pred[i] == min(Y_pred))
  }
  tau_a~ dgamma(0.1, 0.1)
  tau_b ~ dgamma(0.1, 0.1)
  tau_e ~ dgamma(0.1, 0.1)
  sigma2_a <- 1/tau_a
  sigma2_b <- 1/tau_b
  sigma2_e <- 1/tau_e
}

fit_swimmer_model = jags(
  data = list(Y = Y, ns = ns, nt = nt),
  inits = list(list(a = rep(20,4), b = rep(0,4)),
               list(a = rep(30,4), b = rep(1,4)),
               list(a = rep(25,4), b = rep(-1,4)),
               list(a = rep(10,4), b = rep(0,4)),
               list(a = rep(15,4), b = rep(0,4))),

parameters.to.save = c("a","b","sigma2_a","sigma2_b", "sigma2_e", "Y_pred","z"),
  n.chains = 5,
  n.iter = 10000,
  n.burnin = 1000,
  model.file = JAGS_swimmer_model
)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 24
##    Unobserved stochastic nodes: 15
##    Total graph size: 158
##
## Initializing model
```

```
chains = as.mcmc(fit_swimmer_model)
MCMCtrace(chains, pdf = FALSE, params = c("a","b","sigma2_a","sigma2_b", "sigma2_e", "Y_pred"))
```




Trace − a[4]
Density − a[4]
Trace − b[1]
Density − b[1]
Trace − b[2]
Density − b[2]
Trace − b[3]
Density − b[3]
Trace − b[4]
Density − b[4]
Trace − sigma2_a
Density − sigma2_a
Value
Iteration
Density
Parameter estimate


Trace − sigma2_b
Density − sigma2_b
Trace − sigma2_e
Density − sigma2_e
Trace − Y_pred[1]
Density − Y_pred[1]
Trace − Y_pred[2]
Density − Y_pred[2]
Trace − Y_pred[3]
Density − Y_pred[3]
Trace − Y_pred[4]
Density − Y_pred[4]
Value
Iteration
Density
Parameter estimate

```r
gelman.diag(chains, multivariate = FALSE)
```

```
## Potential scale reduction factors:
##
##            Point est. Upper C.I.
## Y_pred[1]        1.00       1.00
## Y_pred[2]        1.00       1.00
## Y_pred[3]        1.00       1.01
## Y_pred[4]        1.00       1.01
## a[1]             1.00       1.00
## a[2]             1.00       1.00
## a[3]             1.00       1.00
## a[4]             1.00       1.00
## b[1]             1.00       1.00
## b[2]             1.00       1.00
## b[3]             1.00       1.00
## b[4]             1.00       1.00
## deviance         1.00       1.01
## sigma2_a         1.01       1.01
## sigma2_b         1.09       1.10
## sigma2_e         1.00       1.00
## z[1]             1.00       1.00
## z[2]              NaN        NaN
## z[3]             1.00       1.00
## z[4]             1.06       1.06
```

## b. PPD

To obtain PPD, draw Y_pred in the JAGS model, densityplots shown for Y_pred is the posterior predictive distribution. We also obtain mean and 95% CI for Y_pred.

```r
summary(chains)
```

```
##
## Iterations = 1:8992
## Thinning interval = 9
## Number of chains = 5
## Sample size per chain = 1000
##
## 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]  22.63421 0.21431 0.0030308      0.0031274
## Y_pred[2]  23.57707 0.20651 0.0029205      0.0028497
## Y_pred[3]  22.91268 0.21203 0.0029986      0.0029293
## Y_pred[4]  23.38329 0.20896 0.0029551      0.0029547
## a[1]       23.22786 0.14225 0.0020117      0.0019784
## a[2]       23.10830 0.14413 0.0020383      0.0020386
## a[3]       22.61924 0.14266 0.0020175      0.0020678
## a[4]       23.73894 0.14266 0.0020175      0.0020173
## b[1]       -0.08508 0.03608 0.0005102      0.0004979
## b[2]        0.06775 0.03614 0.0005111      0.0005176
## b[3]        0.04161 0.03659 0.0005174      0.0005210
```

```
## b[4]        -0.05055 0.03612 0.0005108      0.0005108
## deviance   -33.65562 8.04383 0.1137570      0.1135954
## sigma2_a     2.98535 5.00851 0.0708311      0.0708179
## sigma2_b     0.09813 0.14151 0.0020013      0.0020009
## sigma2_e     0.02346 0.02180 0.0003083      0.0003065
## z[1]         0.82980 0.37585 0.0053153      0.0052624
## z[2]         0.00020 0.01414 0.0002000      0.0002000
## z[3]         0.16680 0.37283 0.0052727      0.0052203
## z[4]         0.00320 0.05648 0.0007988      0.0008346
##
## 2. Quantiles for each variable:
##
##                  2.5%       25%       50%       75%     97.5%
## Y_pred[1]  22.220277  22.50143  22.63117  22.76883  23.04825
## Y_pred[2]  23.172336  23.44412  23.57878  23.71040  23.98216
## Y_pred[3]  22.503286  22.77686  22.91214  23.04770  23.31816
## Y_pred[4]  22.971321  23.24831  23.38051  23.51728  23.79680
## a[1]       22.953355  23.14013  23.22810  23.31931  23.50335
## a[2]       22.822348  23.02151  23.11209  23.20047  23.38406
## a[3]       22.345367  22.52860  22.61770  22.71080  22.90192
## a[4]       23.452953  23.65010  23.74043  23.82952  24.01211
## b[1]       -0.156986  -0.10829  -0.08505  -0.06244  -0.01451
## b[2]       -0.001405   0.04518   0.06733   0.08976   0.13957
## b[3]       -0.029914   0.01772   0.04159   0.06588   0.11344
## b[4]       -0.120128  -0.07376  -0.05106  -0.02755   0.01954
## deviance  -46.608882 -39.19464 -34.48124 -28.93283 -16.36224
## sigma2_a    0.538198   1.13404   1.80302   3.09261  12.18865
## sigma2_b    0.019433   0.03938   0.06230   0.10741   0.40388
## sigma2_e    0.011224   0.01666   0.02096   0.02703   0.04622
## z[1]        0.000000   1.00000   1.00000   1.00000   1.00000
## z[2]        0.000000   0.00000   0.00000   0.00000   0.00000
## z[3]        0.000000   0.00000   0.00000   0.00000   1.00000
## z[4]        0.000000   0.00000   0.00000   0.00000   0.00000
```

### c.

In the JAGS model, we define Z as indicator function that individual i has the fastest posterior predictive value. In this case, we output the posterior mean of Z, as the probablity $Pr(Y_i^* = min(Y_1^*, ..., Y_4^*)|\boldsymbol{Y})$. Based on posterior mean of z = (0.8364, 0.0012, 0.1594, 0.003), we would recommend swimmer 1.

## Problem 2.

### a.

Linear regression.

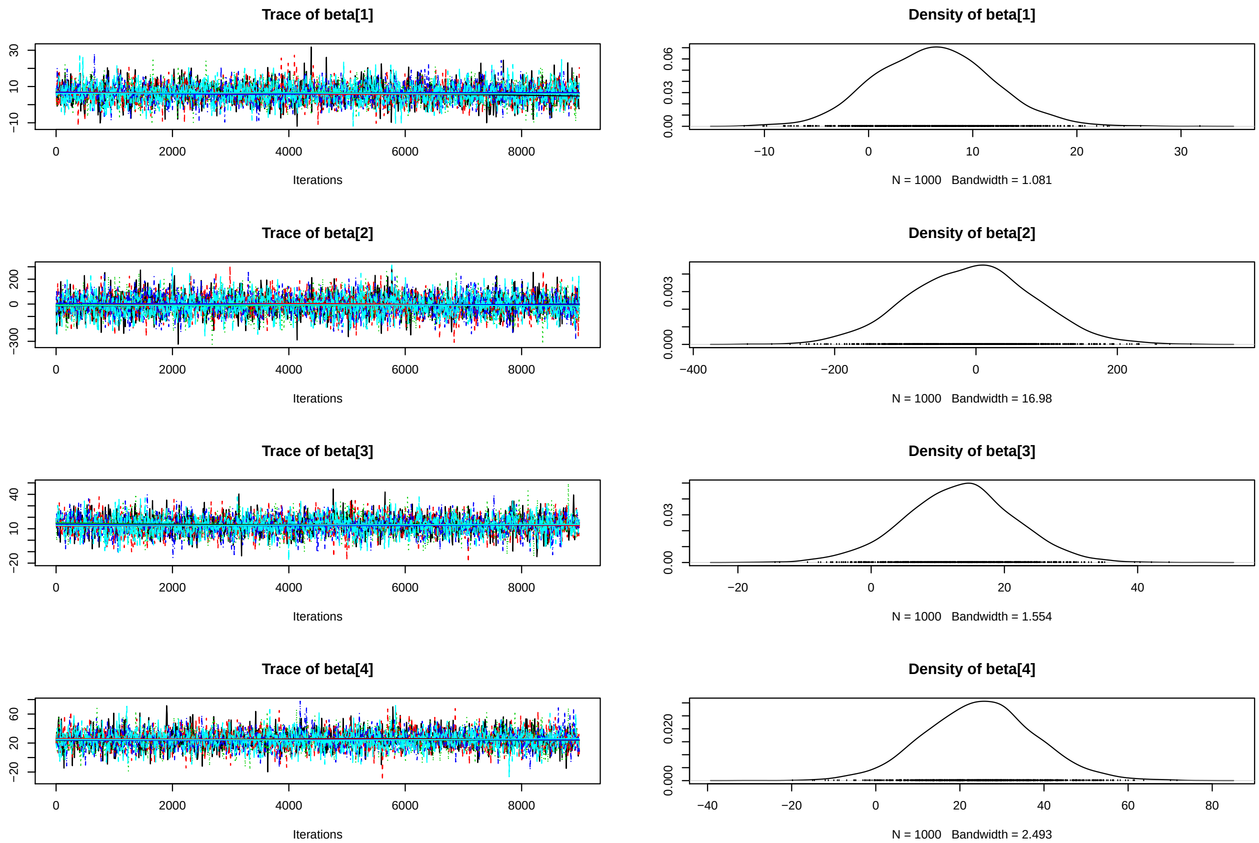
```r
#import data
X = UScrime[,1:15]
Y = UScrime[,16]
n = nrow(UScrime)
p = ncol(UScrime) - 1
df = list()
```

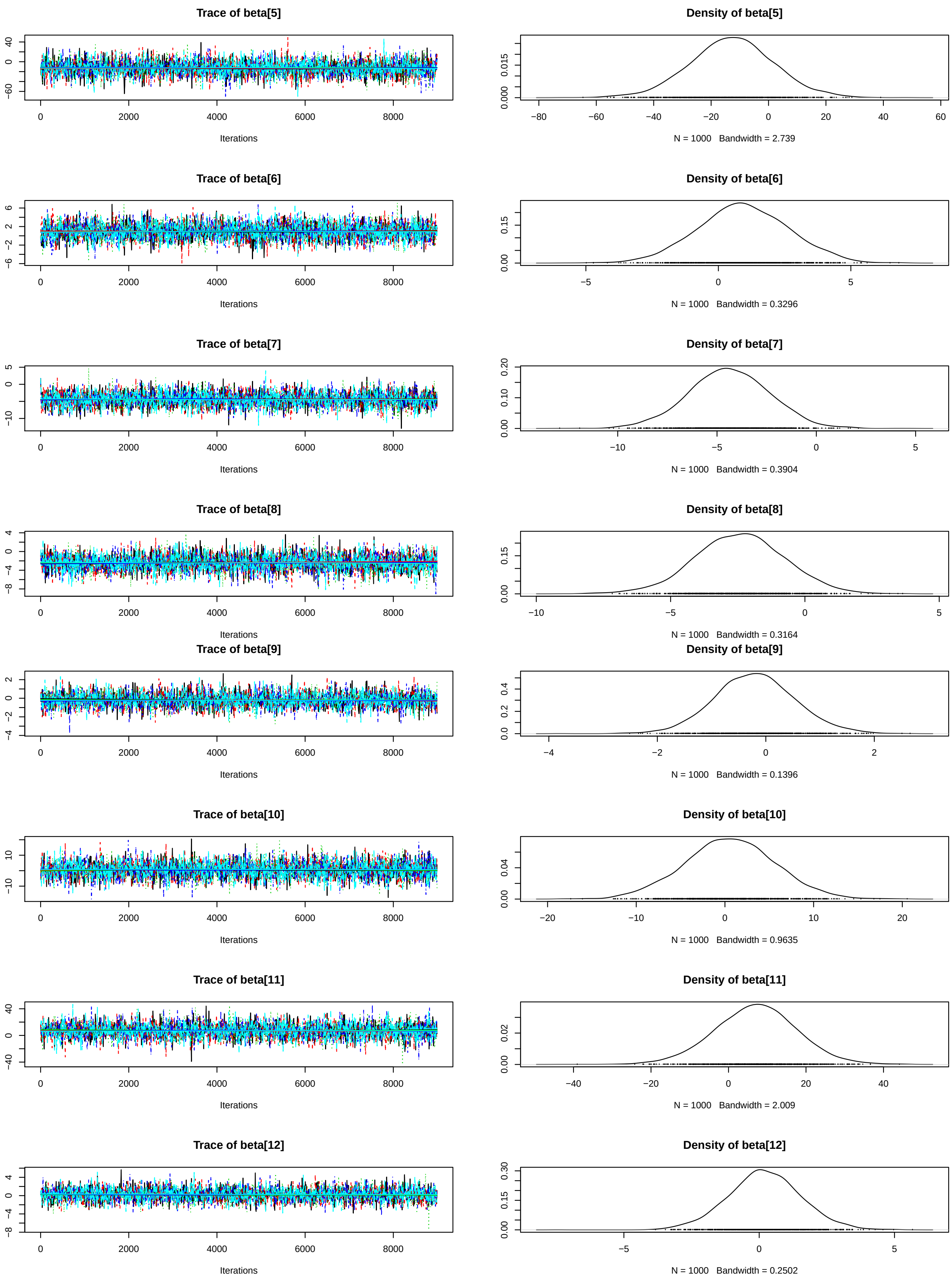
```
df$X = X
df$Y = Y
df$n = n
df$p = p

JAGS_BLR_flat = function(){
  # Likelihood
  for(i in 1:n){
    Y[i] ~ dnorm(mu[i],inv_sigma2)
    mu[i] <- beta_0 + inprod(X[i,],beta)
  }
  # Prior for beta
  for(j in 1:p){
    beta[j] ~ dnorm(0,0.0001)
    #non-informative priors
  }
  # Prior for intercept
  beta_0 ~ dnorm(0, 0.0001)

  # Prior for the inverse variance
  inv_sigma2 ~ dgamma(0.0001, 0.0001)
  sigma2 <- 1.0/inv_sigma2
}


fit_JAGS_flat = jags(data=df,
                inits=list(list(beta = rnorm(p),
                                beta_0 = 0,
                                inv_sigma2 = 1),
                         list(beta = rnorm(p),
                                beta_0 = 1,
                                inv_sigma2 = 2),
                          list(beta = rnorm(p),
                                beta_0 = 2,
                                inv_sigma2 = 2),
                          list(beta = rnorm(p),
                                beta_0 = 10,
                                inv_sigma2 = 5),
                         list(beta = rnorm(p),
                                beta_0 = 20,
                                inv_sigma2 = 1)),
                parameters.to.save = c("beta_0","beta","sigma2"),
                n.chains=5,
                n.iter=10000,
                n.burnin=1000,
                model.file=JAGS_BLR_flat)
```

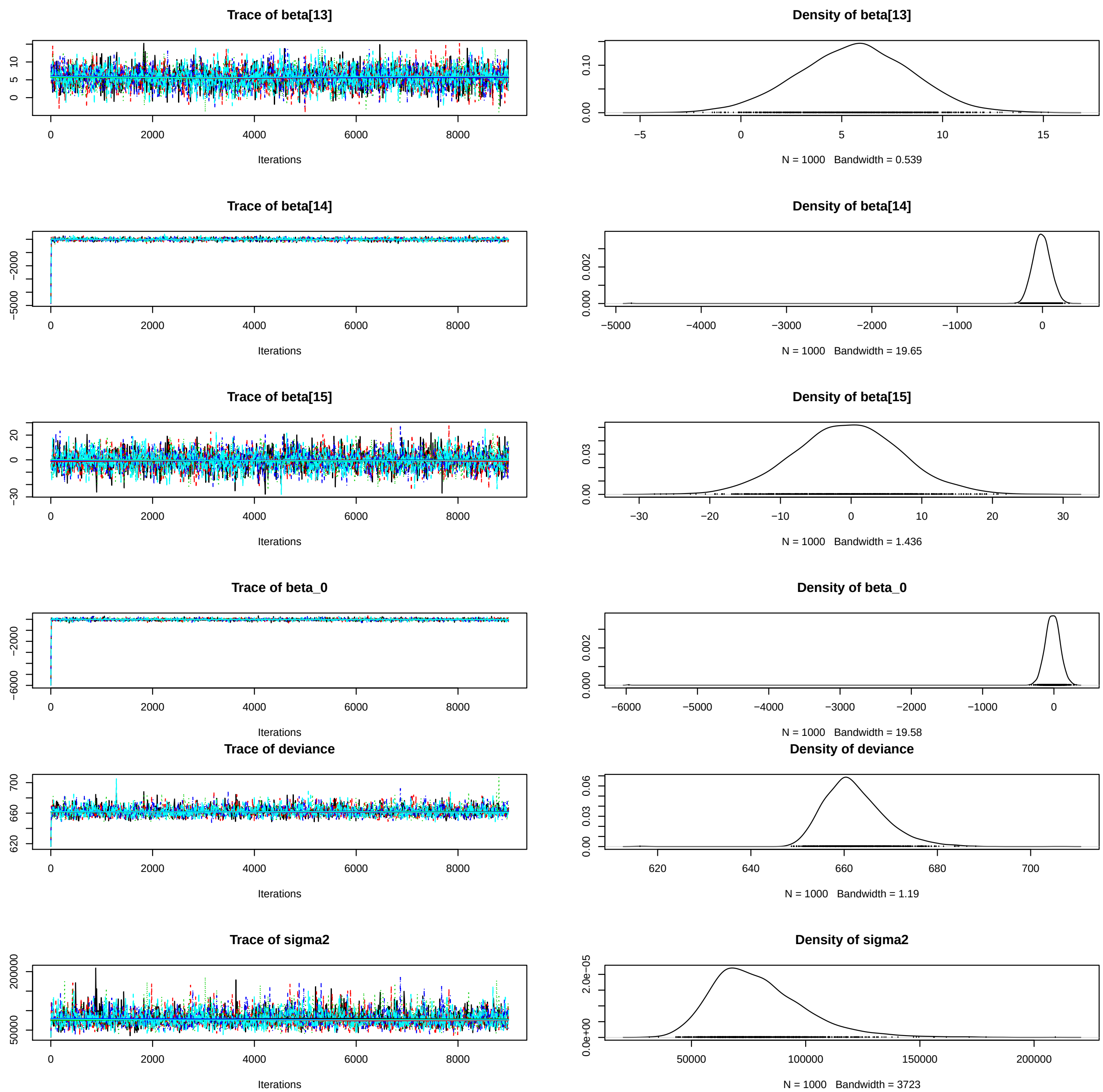
```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 47
##    Unobserved stochastic nodes: 17
##    Total graph size: 950
```

```
##
## Initializing model
```

```
chains_2a = as.mcmc(fit_JAGS_flat)
plot(chains_2a)
```




Trace of beta[5]
Iterations
Density of beta[5]
N = 1000 Bandwidth = 2.739
Trace of beta[6]
Iterations
Density of beta[6]
N = 1000 Bandwidth = 0.3296
Trace of beta[7]
Iterations
Density of beta[7]
N = 1000 Bandwidth = 0.3904
Trace of beta[8]
Iterations
Density of beta[8]
N = 1000 Bandwidth = 0.3164
Trace of beta[9]
Iterations
Density of beta[9]
N = 1000 Bandwidth = 0.1396
Trace of beta[10]
Iterations
Density of beta[10]
N = 1000 Bandwidth = 0.9635
Trace of beta[11]
Iterations
Density of beta[11]
N = 1000 Bandwidth = 2.009
Trace of beta[12]
Iterations
Density of beta[12]
N = 1000 Bandwidth = 0.2502




```r
gelman.diag(chains_2a)
```

```
## Potential scale reduction factors:
##
##           Point est. Upper C.I.
## beta[1]        1.003       1.01
## beta[2]        0.999       1.00
## beta[3]        1.002       1.01
## beta[4]        1.001       1.00
## beta[5]        1.001       1.01
## beta[6]        1.000       1.00
## beta[7]        1.001       1.00
## beta[8]        1.001       1.00
## beta[9]        1.001       1.00
## beta[10]       1.000       1.00
```

```
## beta[11]      1.000       1.00
## beta[12]      1.001       1.00
## beta[13]      1.001       1.00
## beta[14]      1.000       1.00
## beta[15]      1.000       1.00
## beta_0        1.000       1.00
## deviance      1.001       1.00
## sigma2        1.005       1.01
##
## Multivariate psrf
##
## 1.01
```

```
##output 95% credible interval
summary(chains_2a)
```

```
##
## Iterations = 1:8992
## Thinning interval = 9
## Number of chains = 5
## Sample size per chain = 1000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                Mean        SD  Naive SE Time-series SE
## beta[1]      6.2665     5.600   0.07919        0.07773
## beta[2]     -6.2477    87.991   1.24438        1.22961
## beta[3]     13.2462     8.368   0.11835        0.11490
## beta[4]     24.8543    13.238   0.18721        0.18716
## beta[5]    -12.7524    14.686   0.20769        0.20759
## beta[6]      0.9395     1.712   0.02421        0.02328
## beta[7]     -4.3958     2.046   0.02894        0.02829
## beta[8]     -2.4215     1.678   0.02373        0.02372
## beta[9]     -0.1999     0.745   0.01054        0.01004
## beta[10]     0.5547     5.136   0.07264        0.07169
## beta[11]     7.0196    10.712   0.15148        0.15147
## beta[12]     0.2036     1.353   0.01913        0.01952
## beta[13]     5.6415     2.805   0.03966        0.03790
## beta[14]   -13.6725   182.689   2.58361        2.58445
## beta[15]    -0.4990     7.564   0.10696        0.10700
## beta_0     -23.4653   213.148   3.01437        3.01544
## deviance   662.2543     6.606   0.09343        0.09352
## sigma2   79533.4483 20590.852 291.19863      290.99404
##
## 2. Quantiles for each variable:
##
##                2.5%        25%        50%        75%      97.5%
## beta[1]  -4.327e+00  2.391e+00     6.2105     9.9408  1.747e+01
## beta[2]  -1.797e+02 -6.666e+01    -4.6218    51.5321  1.662e+02
## beta[3]  -3.784e+00  7.825e+00    13.4183    18.6130  2.945e+01
## beta[4]  -9.218e-01  1.599e+01    24.8460    33.3070  5.166e+01
## beta[5]  -4.161e+01 -2.221e+01   -12.7385    -3.1869  1.700e+01
## beta[6]  -2.428e+00 -1.824e-01     0.9265     2.1062  4.268e+00
## beta[7]  -8.468e+00 -5.753e+00    -4.4165    -3.0418 -4.739e-01
```

```
## beta[8]  -5.781e+00 -3.527e+00    -2.4091    -1.3297  8.129e-01
## beta[9]  -1.640e+00 -6.926e-01    -0.2052     0.2767  1.312e+00
## beta[10] -9.464e+00 -2.802e+00     0.5451     3.8889  1.066e+01
## beta[11] -1.439e+01 -5.498e-02     7.2358    13.8972  2.812e+01
## beta[12] -2.517e+00 -6.659e-01     0.1811     1.0713  2.925e+00
## beta[13]  9.149e-02  3.803e+00     5.7142     7.5454  1.104e+01
## beta[14] -2.036e+02 -7.807e+01   -10.3631    58.3941  1.889e+02
## beta[15] -1.551e+01 -5.458e+00    -0.4843     4.5134  1.476e+01
## beta_0   -2.134e+02 -8.536e+01   -16.9771    50.5969  1.781e+02
## deviance  6.519e+02  6.577e+02   661.4326   666.0076  6.771e+02
## sigma2    4.852e+04  6.488e+04 76334.0844 90729.4792  1.284e+05
```

## b.

Cross validation

```
#split data into training and test set
split_data = function(df,train_test_ratio = 1,random=TRUE){
  n_train = floor(df$n*train_test_ratio/(1+train_test_ratio))
  n_test  =  df$n - n_train
  if(random){
    train_idx = sample(1:n,n_train,replace = FALSE)
    test_idx = setdiff(1:n,train_idx)
  }
  else{
    train_idx = 1:n_train
    test_idx = n_train+1:n_test
  }

  df_t = list()
  df_t$Y_train = df$Y[train_idx]
  df_t$X_train = df$X[train_idx,,drop=FALSE]
  df_t$X_test = df$X[test_idx,,drop=FALSE]
  df_t$n_train = n_train
  df_t$n_test = n_test
  df_t$p = df$p

  return(list(df_t=df_t,Y_test=df$Y[test_idx]))
}

pred = split_data(df, random = FALSE)

##define a predictive JAGS
JAGS_BLR_flat_pred = function(){
  # Likelihood
  for(i in 1:n_train){
    Y_train[i] ~ dnorm(mu_train[i],inv_sigma2)
    mu_train[i] <- beta_0 + inprod(X_train[i,],beta)
    # same as beta_0 + X[i,1]*beta[1] + ... + X[i,p]*beta[p]
  }
  # Prior for beta
  for(j in 1:p){
    beta[j] ~ dnorm(0,0.0001)
    #non-informative priors
```

```
  }
  # Prior for intercept
  beta_0 ~ dnorm(0, 0.0001)

  # Prior for the inverse variance
  inv_sigma2 ~ dgamma(0.0001, 0.0001)
  sigma2 <- 1.0/inv_sigma2

  #prediction
   # Predictions
  for(i in 1:n_test){
    Y_test[i] ~ dnorm(mu_test[i],inv_sigma2)
    mu_test[i] <- beta_0 + inprod(X_test[i,],beta)
  }
}

fit_JAGS_flat_pred = jags(data=pred$df_t,
                inits=list(list(beta = rnorm(p),
                                beta_0 = 0,
                                inv_sigma2 = 1),
                         list(beta = rnorm(p),
                                beta_0 = 1,
                                inv_sigma2 = 2),
                          list(beta = rnorm(p),
                                beta_0 = 2,
                                inv_sigma2 = 2),
                          list(beta = rnorm(p),
                                beta_0 = 10,
                                inv_sigma2 = 5),
                         list(beta = rnorm(p),
                                beta_0 = 20,
                                inv_sigma2 = 1)),
                parameters.to.save = c("beta_0","beta","sigma2", "Y_test"),
                n.chains=5,
                n.iter=10000,
                n.burnin=1000,
                model.file=JAGS_BLR_flat_pred)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 23
##    Unobserved stochastic nodes: 41
##    Total graph size: 951
##
## Initializing model
```

```
chains_2b = as.mcmc(fit_JAGS_flat_pred)

##plot the predictive values
result = summary(chains_2b)
q = result$quantiles
dtfr = as.data.frame(cbind(result$quantiles))
```

```r
##compare
pred$Y_test
```

```
##  [1]  968  523 1993  342 1216 1043  696  373  754 1072  923  653 1272  831
## [15]  566  826 1151  880  542  823 1030  455  508  849
```

```r
fit_JAGS_flat_pred$BUGSoutput$median$Y_test
```

```
##  [1]  702.3189  911.1052 1694.7808  190.2616  861.1977 2098.1610  798.2761
##  [8]  289.2185 1290.1620  658.4909  761.7653 1272.8585  764.8326  562.4828
## [15]  745.2396  995.5814 1403.8654  534.4566  503.4575  887.8674  580.6554
## [22]  550.1389 1331.0586  592.6950
```

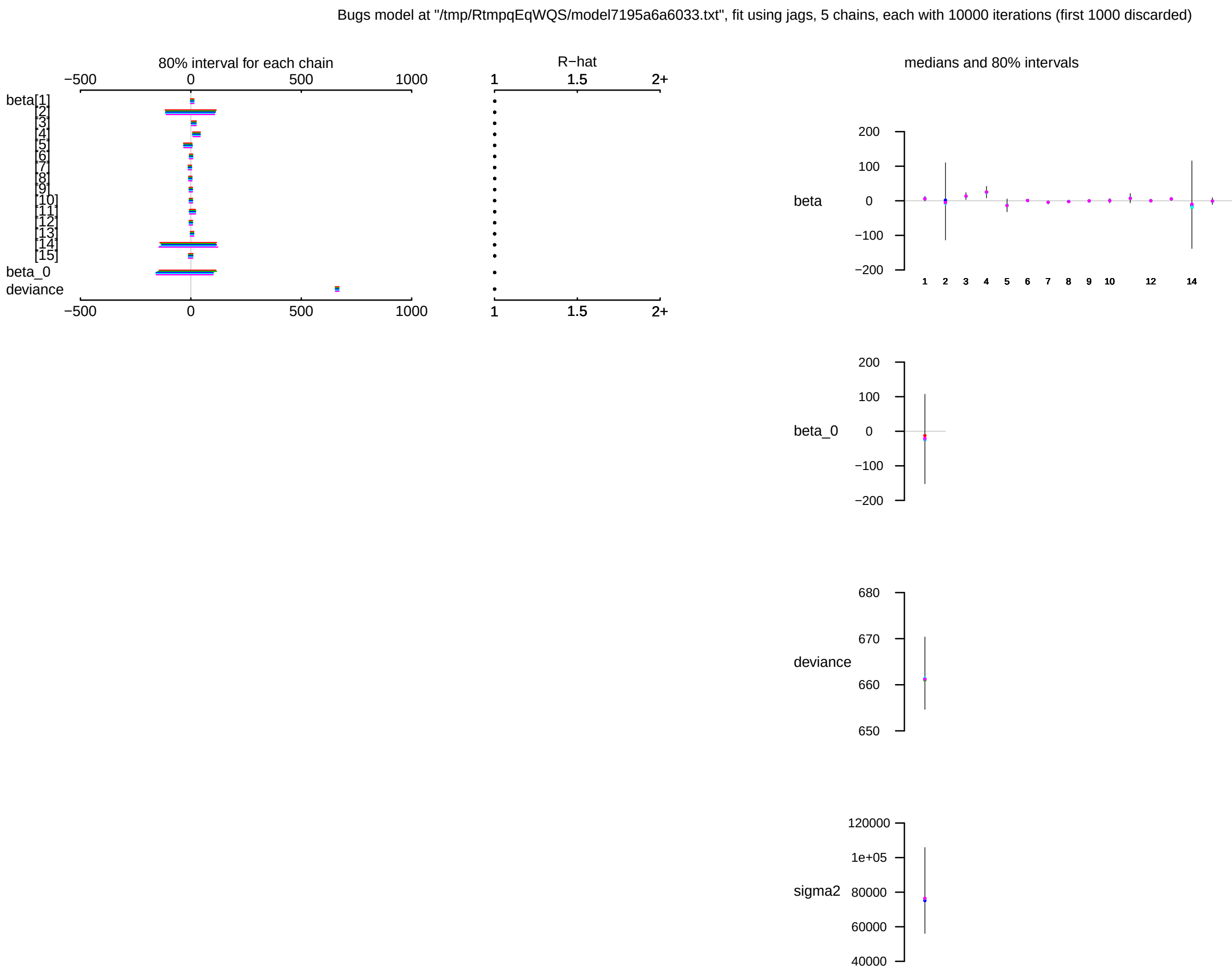
### c. Slab and spike

```r
JAGS_BLR_SpikeSlab = function(){
  # Likelihood
  for(i in 1:n){
    Y[i] ~ dnorm(mu[i],inv_sigma2)
    mu[i] <- beta_0 + inprod(X[i,],beta)
    # same as beta_0 + X[i,1]*beta[1] + ... + X[i,p]*beta[p]
  }
  #Prior for beta_j
  for(j in 1:p){
    beta[j] ~ dnorm(0,inv_tau2[j])
    inv_tau2[j] <- (1-gamma[j])*1000+gamma[j]*0.01
    gamma[j] ~ dbern(0.5)
  }
  # Prior for intercept
  beta_0 ~ dnorm(0, 0.0001)

  # Prior for the inverse variance
  inv_sigma2 ~ dgamma(0.0001, 0.0001)
  sigma2 <- 1.0/inv_sigma2
}

fit_JAGS_SpikeSlab = jags(data=df,
                inits=list(list(beta = rnorm(p),
                                beta_0 = 0,
                                inv_sigma2 = 1),
                         list(beta = rnorm(p),
                                beta_0 = 1,
                                inv_sigma2 = 2),
                          list(beta = rnorm(p),
                                beta_0 = 2,
                                inv_sigma2 = 2),
                          list(beta = rnorm(p),
                                beta_0 = 10,
                                inv_sigma2 = 5),
                         list(beta = rnorm(p),
                                beta_0 = 20,
                                inv_sigma2 = 1)),
                parameters.to.save = c("beta_0","beta","sigma2"),
```

```
                 n.chains=5,
                 n.iter=10000,
                 n.burnin=1000,
                 model.file=JAGS_BLR_flat)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 47
##    Unobserved stochastic nodes: 17
##    Total graph size: 950
##
## Initializing model
```

```
plot(fit_JAGS_SpikeSlab)
```




```
chains_2c = as.mcmc(fit_JAGS_SpikeSlab)
summary(chains_2c)
```

```
##
## Iterations = 1:8992
## Thinning interval = 9
## Number of chains = 5
```

```
## Sample size per chain = 1000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                 Mean        SD  Naive SE Time-series SE
## beta[1]       6.3092     5.400   0.07637        0.07628
## beta[2]      -2.6117    88.419   1.25043        1.25034
## beta[3]      13.2802     8.259   0.11680        0.11681
## beta[4]      25.1446    13.178   0.18636        0.18134
## beta[5]     -13.0849    14.577   0.20615        0.20049
## beta[6]       0.9506     1.692   0.02392        0.02419
## beta[7]      -4.3264     1.991   0.02815        0.02816
## beta[8]      -2.3521     1.654   0.02339        0.02338
## beta[9]      -0.2120     0.739   0.01045        0.01053
## beta[10]      0.3355     5.068   0.07167        0.07170
## beta[11]      7.4999    10.496   0.14843        0.14843
## beta[12]      0.1314     1.325   0.01874        0.01846
## beta[13]      5.4855     2.782   0.03934        0.03935
## beta[14]    -17.3998   182.078   2.57497        2.57593
## beta[15]     -0.7892     7.525   0.10642        0.10578
## beta_0      -27.1134   213.578   3.02045        3.02106
## deviance    662.0127     6.519   0.09219        0.09116
## sigma2    79143.6611 20658.342 292.15307      306.47839
##
## 2. Quantiles for each variable:
##
##                 2.5%        25%        50%        75%      97.5%
## beta[1]   -4.554e+00     2.7418     6.3351     9.9140  1.670e+01
## beta[2]   -1.759e+02   -61.6151    -4.4024    55.8237  1.725e+02
## beta[3]   -3.055e+00     7.8237    13.3335    18.6867  2.982e+01
## beta[4]    1.121e-01    16.3637    25.0514    33.8903  5.097e+01
## beta[5]   -4.198e+01   -22.7576   -13.1229    -3.4576  1.527e+01
## beta[6]   -2.346e+00    -0.1727     0.9546     2.0783  4.264e+00
## beta[7]   -8.164e+00    -5.6202    -4.3302    -3.0469 -4.070e-01
## beta[8]   -5.645e+00    -3.4453    -2.3590    -1.2744  9.474e-01
## beta[9]   -1.653e+00    -0.6958    -0.1988     0.2833  1.220e+00
## beta[10]  -9.754e+00    -3.0409     0.4276     3.7375  1.029e+01
## beta[11]  -1.295e+01     0.6334     7.4273    14.4106  2.804e+01
## beta[12]  -2.517e+00    -0.7261     0.1227     1.0188  2.759e+00
## beta[13]   9.279e-02     3.5854     5.4443     7.3662  1.102e+01
## beta[14]  -2.079e+02   -80.9641   -14.8048    53.8219  1.900e+02
## beta[15]  -1.534e+01    -5.6004    -0.9122     4.1580  1.398e+01
## beta_0    -2.216e+02   -89.2852   -19.9943    46.1587  1.732e+02
## deviance   6.518e+02   657.5149   661.2607   665.6451  6.771e+02
## sigma2     4.816e+04 64411.3123 76117.5539 89902.4117  1.282e+05
```

```
###############################################################################
##Prediction
###############################################################################


JAGS_BLR_SpikeSlab_pred = function(){
  # Likelihood
```

```
  for(i in 1:n_train){
    Y_train[i] ~ dnorm(mu_train[i],inv_sigma2)
    mu_train[i] <- beta_0 + inprod(X_train[i,],beta)
    # same as beta_0 + X[i,1]*beta[1] + ... + X[i,p]*beta[p]
  }
  # Prior for beta
  for(j in 1:p){
    beta[j] ~ dnorm(0,inv_tau2[j])
    inv_tau2[j] <- (1-gamma[j])*1000+gamma[j]*0.01
    gamma[j] ~ dbern(0.5)
  }
  # Prior for intercept
  beta_0 ~ dnorm(0, 0.0001)

  # Prior for the inverse variance
  inv_sigma2 ~ dgamma(0.0001, 0.0001)
  sigma2 <- 1.0/inv_sigma2

  #prediction
   # Predictions
  for(i in 1:n_test){
    Y_test[i] ~ dnorm(mu_test[i],inv_sigma2)
    mu_test[i] <- beta_0 + inprod(X_test[i,],beta)
  }
}

fit_JAGS_SpikeSlab_pred = jags(data = pred$df_t,
                inits=list(list(beta = rnorm(p),
                                beta_0 = 0,
                                inv_sigma2 = 1),
                          list(beta = rnorm(p),
                                beta_0 = 1,
                                inv_sigma2 = 2),
                           list(beta = rnorm(p),
                                beta_0 = 2,
                                inv_sigma2 = 2),
                           list(beta = rnorm(p),
                                beta_0 = 10,
                                inv_sigma2 = 5),
                          list(beta = rnorm(p),
                                beta_0 = 20,
                                inv_sigma2 = 1)),
                parameters.to.save = c("beta_0","beta","sigma2","Y_test"),
                n.chains=5,
                n.iter=10000,
                n.burnin=1000,
                model.file=JAGS_BLR_SpikeSlab_pred)
```

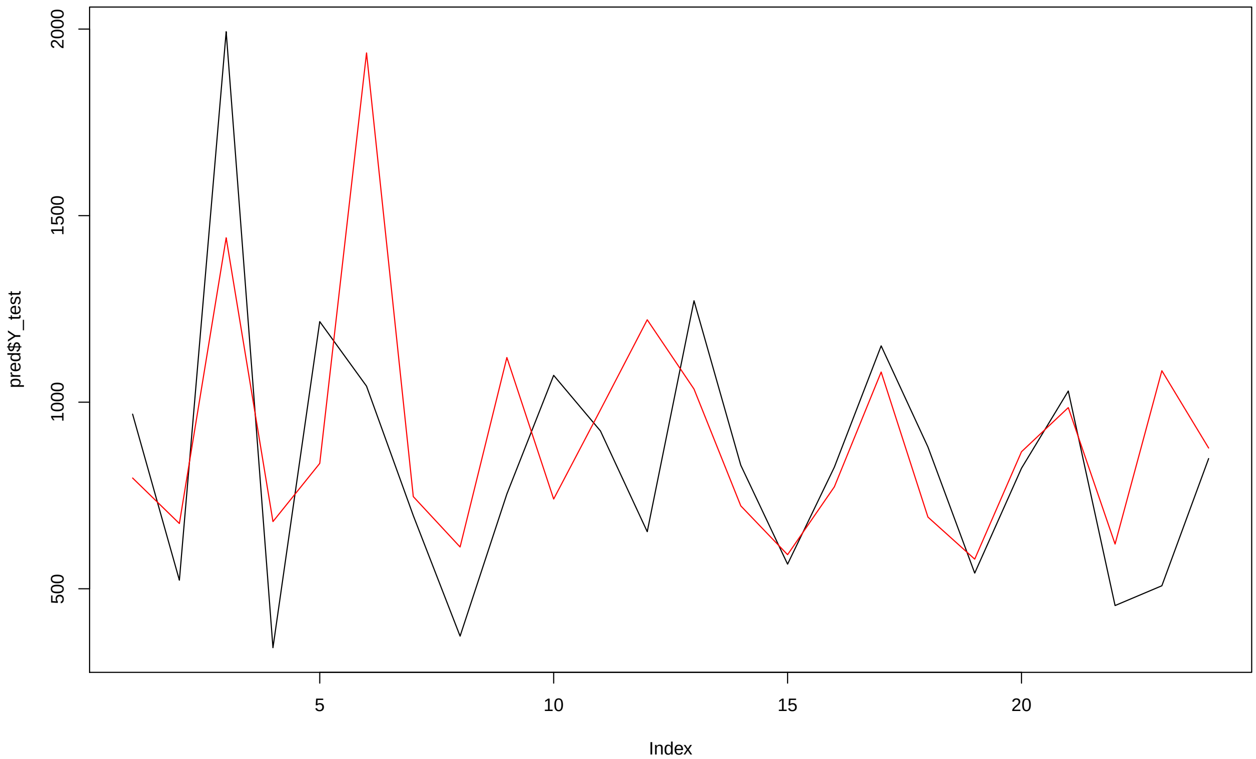
```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 23
##    Unobserved stochastic nodes: 56
```

```
##    Total graph size: 1071
##
## Initializing model
```

```
cbind(pred$Y_test, fit_JAGS_SpikeSlab_pred$BUGSoutput$median$Y_test)
```

```
##       [,1]      [,2]
##  [1,]  968  796.8714
##  [2,]  523  674.8399
##  [3,] 1993 1440.6227
##  [4,]  342  680.0654
##  [5,] 1216  835.9979
##  [6,] 1043 1936.1407
##  [7,]  696  746.9942
##  [8,]  373  611.9618
##  [9,]  754 1119.7948
## [10,] 1072  740.4464
## [11,]  923  979.5285
## [12,]  653 1220.9262
## [13,] 1272 1035.2173
## [14,]  831  722.1332
## [15,]  566  591.3444
## [16,]  826  773.2865
## [17,] 1151 1080.8901
## [18,]  880  692.0879
## [19,]  542  579.3675
## [20,]  823  867.2079
## [21,] 1030  985.5025
## [22,]  455  619.7641
## [23,]  508 1084.4153
## [24,]  849  877.1035
```

```
plot(pred$Y_test, type = 'l')
lines(fit_JAGS_SpikeSlab_pred$BUGSoutput$median$Y_test, col= 'red')
```




```r
v_loc = unique(gambia[,"x"])
v = match(gambia[,"x"],v_loc)
```