# Biostat 682 Homework 5

*David (Daiwei) Zhang*

*December 6, 2017*

## Problem 1

### (a)

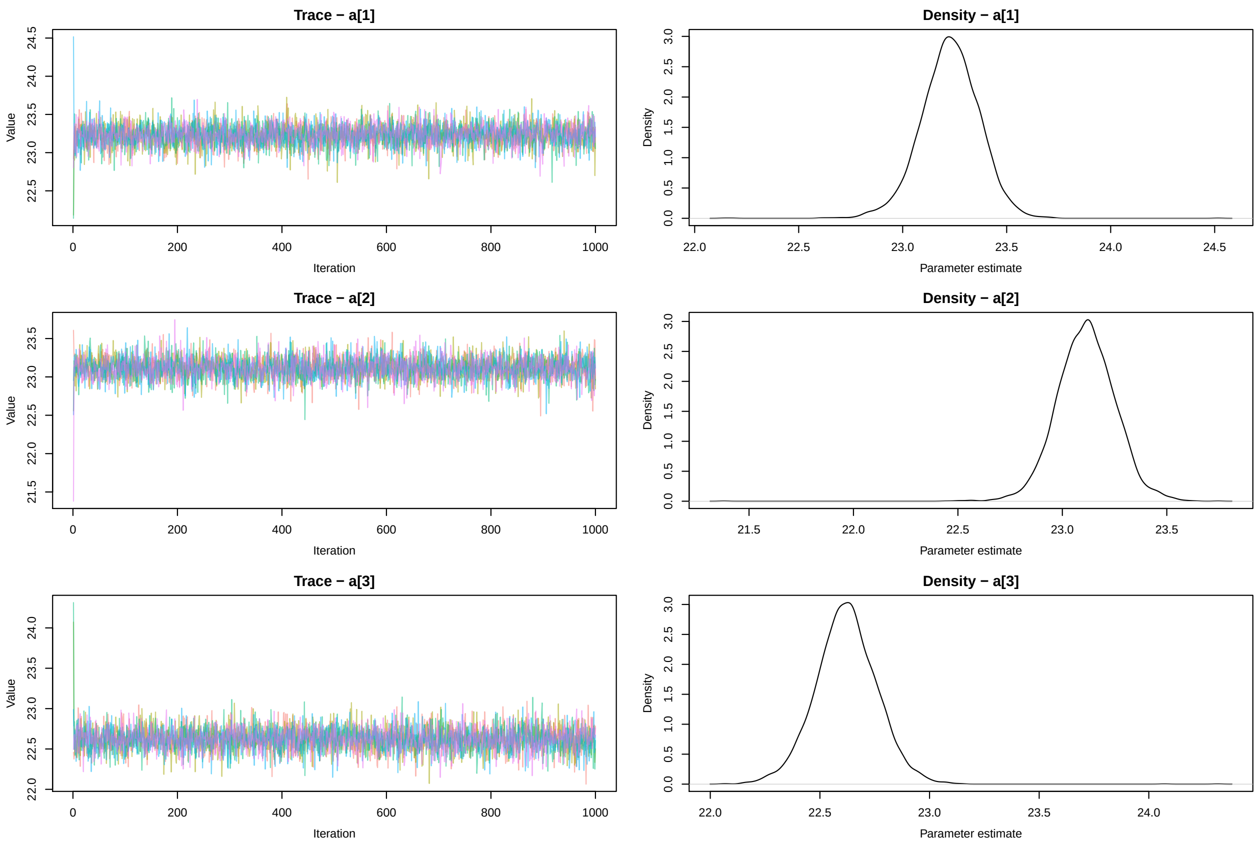
We first fit a Bayesian linear regression model.

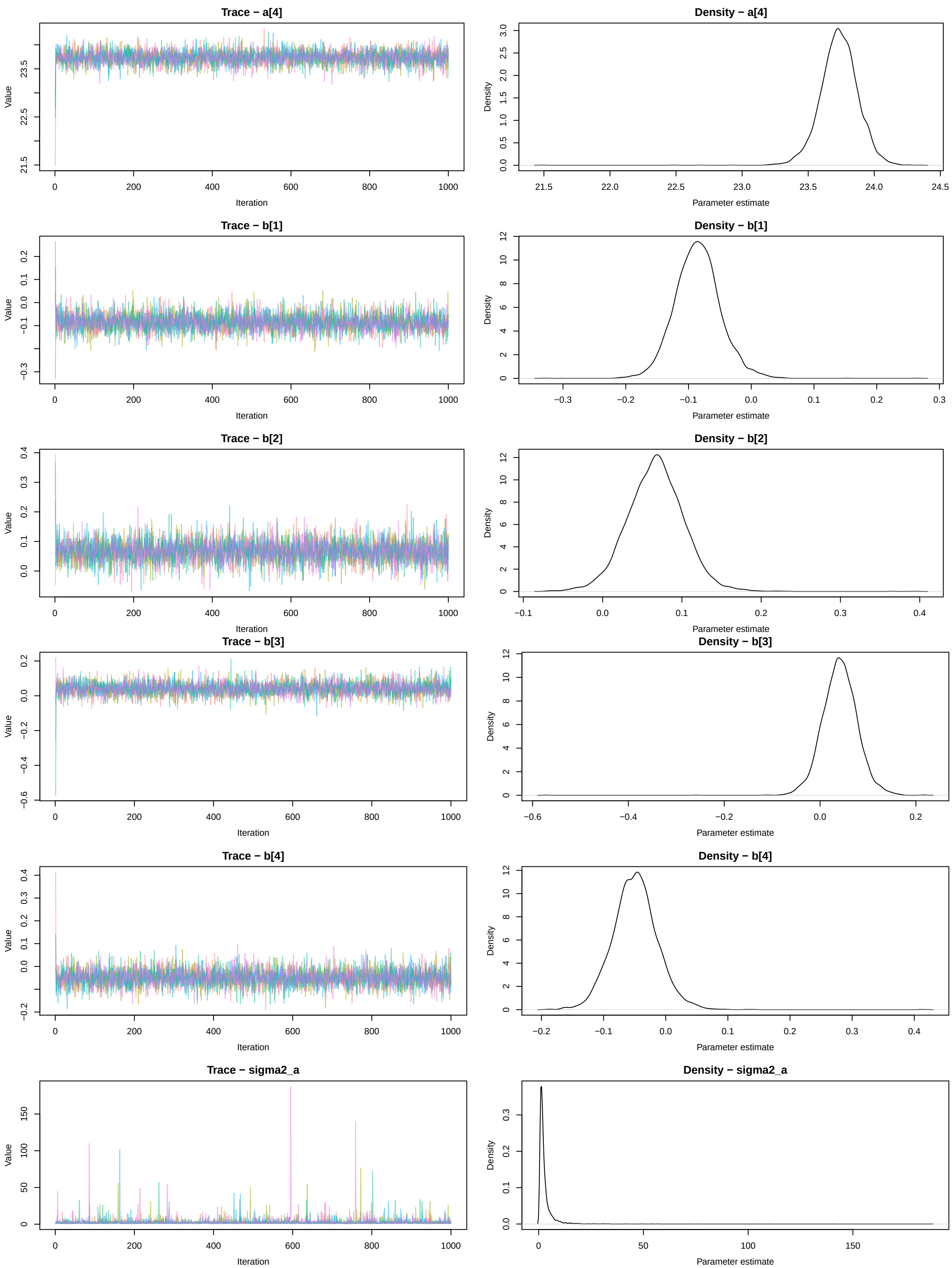
```
load("swim_time.RData")
ns   <- nrow(Y)
nt   <- ncol(Y)
JAGS_swimmer_model = function(){
  for (i in 1:ns) {
    for (j in 1:nt) {
      Y[i, j]    ~ dnorm(mean[i, j], tau_e)
      mean[i, j] <- a[i] + b[i] *j
    }

  }
  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
```

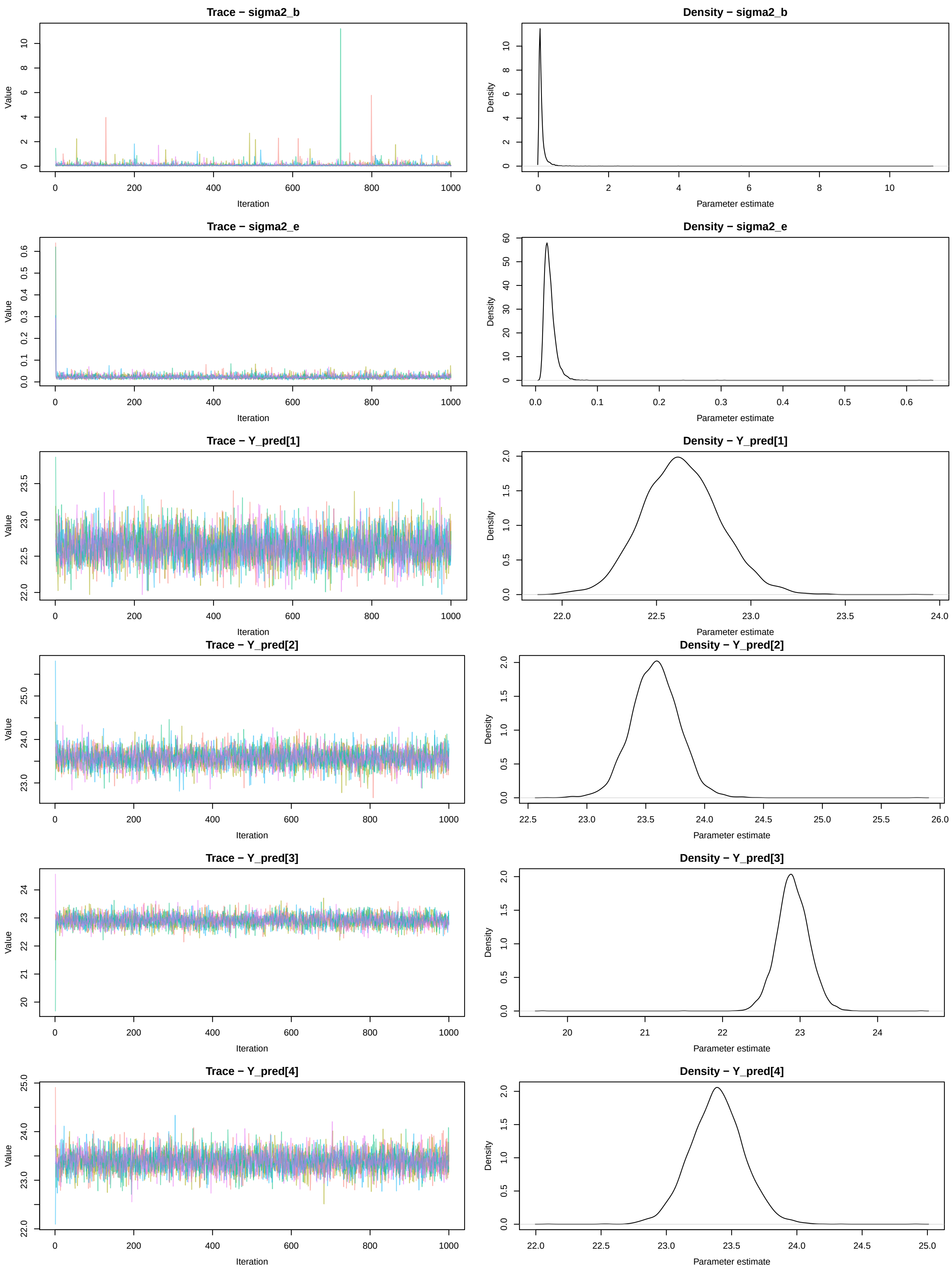
```
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 24
##    Unobserved stochastic nodes: 15
##    Total graph size: 158
##
## Initializing model
```

```r
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]      1.000      1.002
## Y_pred[2]      1.000      1.000
## Y_pred[3]      1.001      1.004
## Y_pred[4]      1.000      1.001
## a[1]           1.001      1.003
## a[2]           1.001      1.004
## a[3]           1.000      1.001
## a[4]           1.000      1.001
## b[1]           1.002      1.005
## b[2]           1.000      1.001
## b[3]           1.000      1.001
## b[4]           0.999      1.000
## deviance       1.000      1.001
## sigma2_a       1.120      1.132
## sigma2_b       1.134      1.141
## sigma2_e       1.004      1.008
## z[1]           0.999      0.999
## z[2]           1.291      1.349
## z[3]           0.999      1.000
## z[4]           1.193      1.221
```

### (b)

Next, we find the predictive posterior distribution for each player two weeks after the last recorded time. This is represented by the Y_pred variable in the JAGS analysis.

```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]  22.63097 0.20529 0.0029032      0.0028966
## Y_pred[2]  23.58072 0.20592 0.0029122      0.0029505
## Y_pred[3]  22.90774 0.21109 0.0029853      0.0029848
## Y_pred[4]  23.38855 0.20862 0.0029503      0.0029513
## a[1]       23.22821 0.14186 0.0020062      0.0020184
## a[2]       23.11350 0.14103 0.0019945      0.0019790
## a[3]       22.62035 0.14338 0.0020277      0.0020510
## a[4]       23.73432 0.14432 0.0020410      0.0020222
## b[1]       -0.08540 0.03645 0.0005154      0.0005154
## b[2]        0.06678 0.03608 0.0005102      0.0005190
## b[3]        0.04120 0.03717 0.0005257      0.0005306
```

```
## b[4]       -0.04941 0.03643 0.0005152      0.0005152
## deviance  -33.67947 7.97645 0.1128041      0.1128108
## sigma2_a    2.95767 5.56274 0.0786690      0.0786558
## sigma2_b    0.10129 0.22864 0.0032335      0.0032128
## sigma2_e    0.02322 0.01624 0.0002297      0.0002298
## z[1]        0.83440 0.37176 0.0052575      0.0050340
## z[2]        0.00040 0.02000 0.0002828      0.0002828
## z[3]        0.16300 0.36940 0.0052241      0.0050184
## z[4]        0.00220 0.04686 0.0006627      0.0007085
##
## 2. Quantiles for each variable:
##
##                 2.5%       25%       50%       75%     97.5%
## Y_pred[1]  22.246890  22.49114  22.62494  22.76372  23.03914
## Y_pred[2]  23.189062  23.44865  23.58047  23.71412  23.97984
## Y_pred[3]  22.507415  22.77482  22.90209  23.04249  23.31647
## Y_pred[4]  22.983546  23.25232  23.38733  23.52321  23.79711
## a[1]       22.941777  23.13937  23.22989  23.31998  23.49473
## a[2]       22.843057  23.02321  23.11488  23.20347  23.38830
## a[3]       22.341695  22.53097  22.61946  22.70975  22.89574
## a[4]       23.448898  23.64801  23.73479  23.82382  24.00199
## b[1]       -0.153804  -0.10913  -0.08595  -0.06354  -0.01182
## b[2]       -0.003734   0.04407   0.06692   0.08947   0.13753
## b[3]       -0.030376   0.01829   0.04108   0.06462   0.11387
## b[4]       -0.118591  -0.07228  -0.04948  -0.02757   0.02430
## deviance  -46.753824 -39.02088 -34.43283 -29.24725 -16.71285
## sigma2_a    0.557106   1.14084   1.78824   3.06269  11.91522
## sigma2_b    0.018851   0.03851   0.06219   0.10614   0.39050
## sigma2_e    0.011300   0.01671   0.02099   0.02681   0.04515
## 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)

The variables z[i] shows the probability that player i will be the fastest two weeks after the last recording. As we can see, player 1 is the most likely to be the fastest (z[1] = 0.83440).

## Problem 2

### (a)

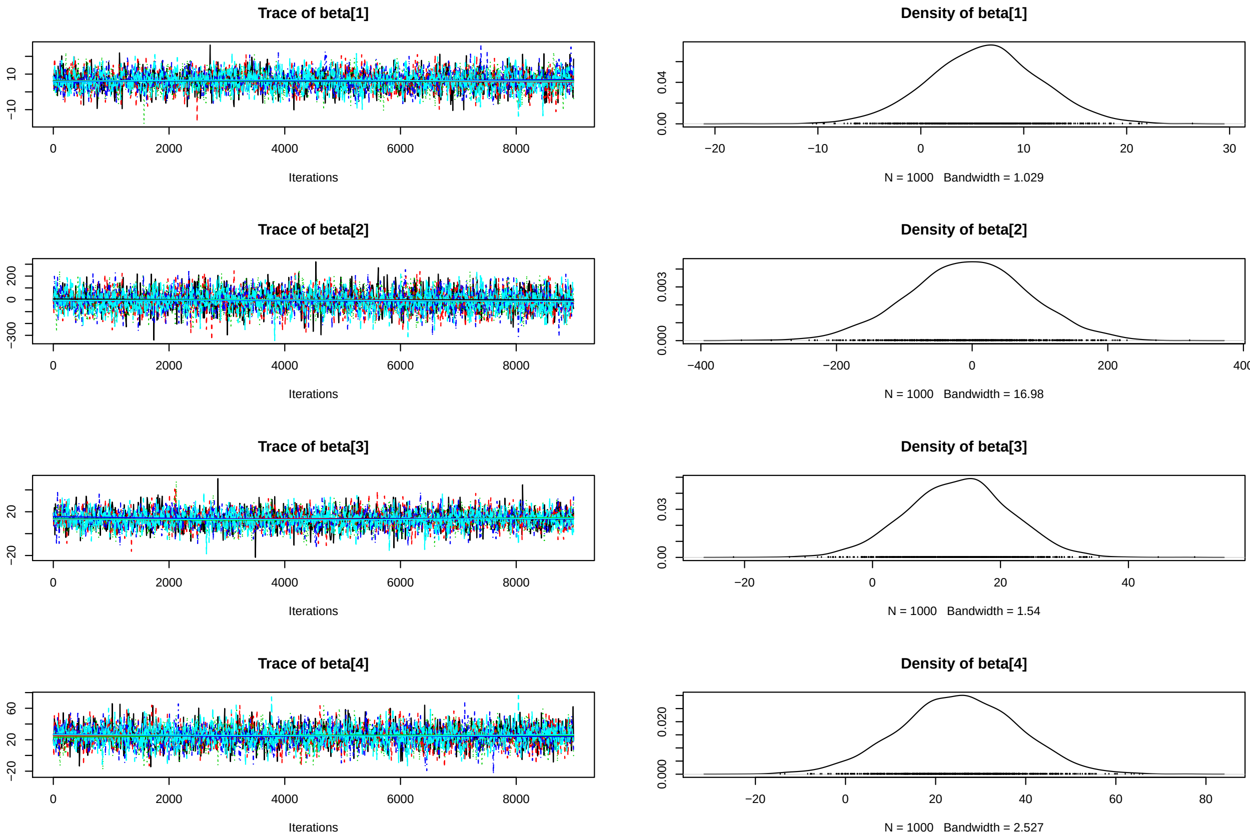
We fit a linear regression model to study the relation between crime rates and explorotary variables.

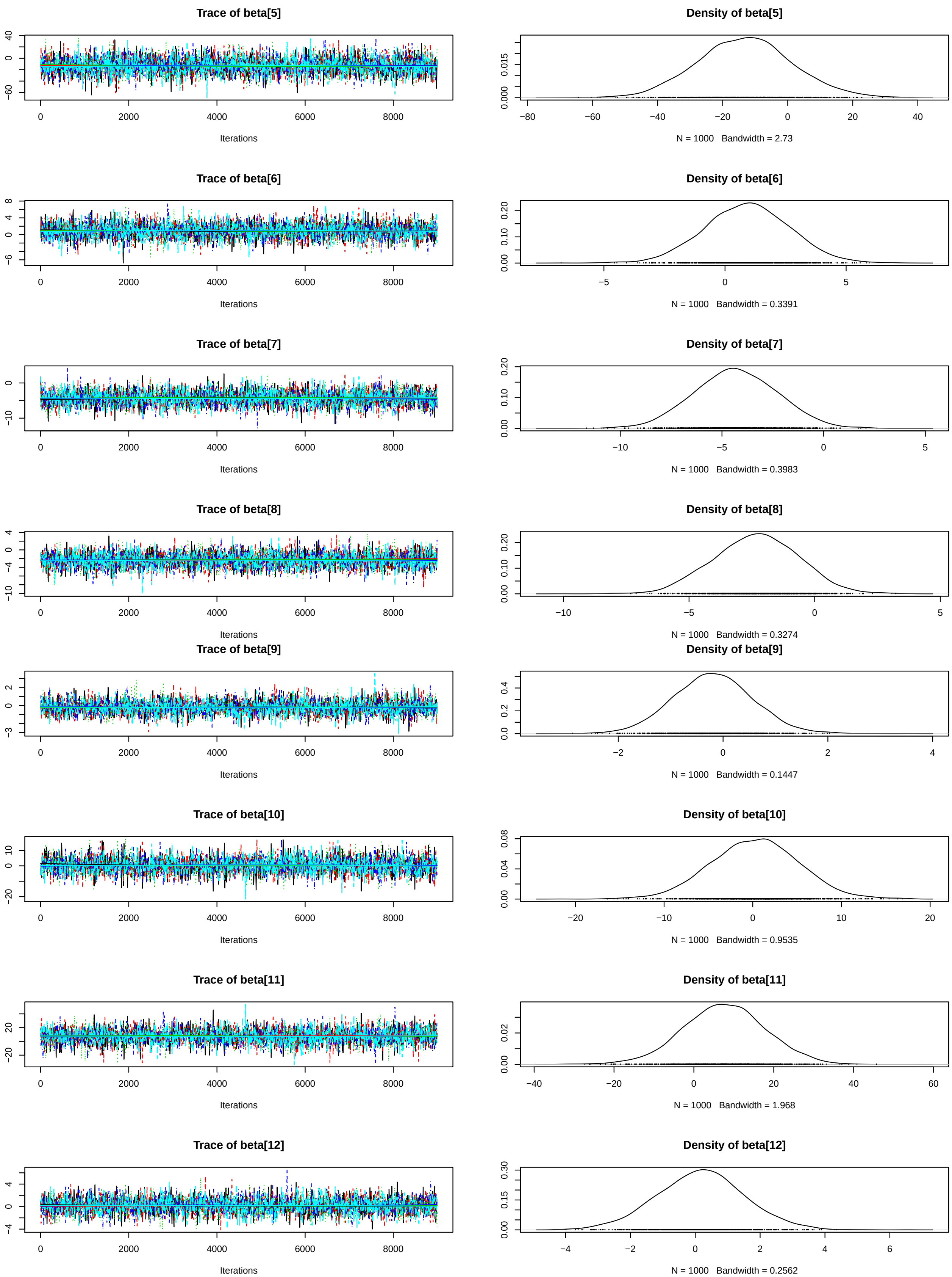
```r
X = UScrime[,1:15]
Y = UScrime[,16]
n = nrow(UScrime)
p = ncol(UScrime) - 1
df = list()
df$X = X
df$Y = Y
df$n = n
```

```r
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)
```

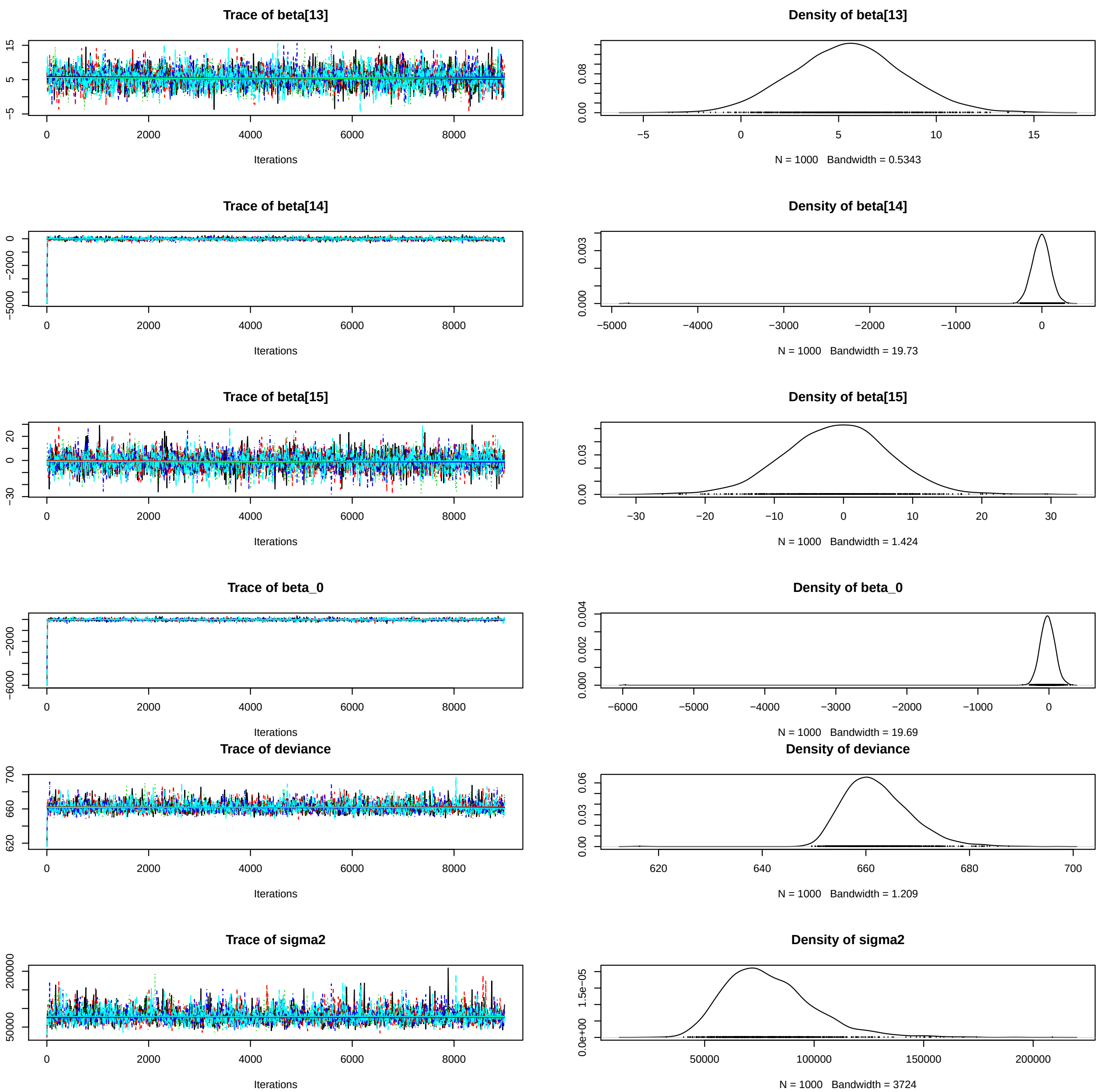
```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 47
##    Unobserved stochastic nodes: 17
##    Total graph size: 950
##
## Initializing model
```

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





Trace of beta[5]
Iterations
Density of beta[5]
N = 1000 Bandwidth = 2.73
Trace of beta[6]
Iterations
Density of beta[6]
N = 1000 Bandwidth = 0.3391
Trace of beta[7]
Iterations
Density of beta[7]
N = 1000 Bandwidth = 0.3983
Trace of beta[8]
Iterations
Density of beta[8]
N = 1000 Bandwidth = 0.3274
Trace of beta[9]
Iterations
Density of beta[9]
N = 1000 Bandwidth = 0.1447
Trace of beta[10]
Iterations
Density of beta[10]
N = 1000 Bandwidth = 0.9535
Trace of beta[11]
Iterations
Density of beta[11]
N = 1000 Bandwidth = 1.968
Trace of beta[12]
Iterations
Density of beta[12]
N = 1000 Bandwidth = 0.2562




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

```
## 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       1.01
## beta[9]           1       1.00
## beta[10]          1       1.00
```

```
## beta[11]          1       1.00
## beta[12]          1       1.00
## beta[13]          1       1.00
## beta[14]          1       1.00
## beta[15]          1       1.00
## beta_0            1       1.00
## deviance          1       1.00
## sigma2            1       1.00
##
## Multivariate psrf
##
## 1
```

```r
##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]      6.1952 5.426e+00   0.07674        0.07965
## beta[2]     -5.6613 8.921e+01   1.26157        1.26140
## beta[3]     13.3626 8.159e+00   0.11539        0.11497
## beta[4]     25.0500 1.315e+01   0.18595        0.18595
## beta[5]    -13.1176 1.459e+01   0.20627        0.20629
## beta[6]      0.9175 1.757e+00   0.02485        0.02539
## beta[7]     -4.3134 2.066e+00   0.02922        0.02963
## beta[8]     -2.3231 1.696e+00   0.02399        0.02407
## beta[9]     -0.1894 7.618e-01   0.01077        0.01108
## beta[10]     0.2607 5.005e+00   0.07078        0.06998
## beta[11]     7.6036 1.047e+01   0.14808        0.14179
## beta[12]     0.1704 1.355e+00   0.01916        0.01814
## beta[13]     5.5164 2.842e+00   0.04019        0.04018
## beta[14]   -12.4536 1.822e+02   2.57650        2.57746
## beta[15]    -0.9172 7.509e+00   0.10620        0.10612
## beta_0     -25.4205 2.134e+02   3.01759        3.01829
## deviance   662.2361 6.508e+00   0.09204        0.09203
## sigma2   79663.7079 2.062e+04 291.62995      301.33489
##
## 2. Quantiles for each variable:
##
##                2.5%        25%        50%        75%      97.5%
## beta[1]  -4.544e+00     2.6322     6.2293     9.7764  1.690e+01
## beta[2]  -1.832e+02   -64.1543    -4.8799    53.7847  1.708e+02
## beta[3]  -2.965e+00     7.9769    13.5693    18.6673  2.901e+01
## beta[4]  -1.285e+00    16.4874    25.1555    34.0370  5.044e+01
## beta[5]  -4.122e+01   -22.7482   -13.1224    -3.7934  1.629e+01
## beta[6]  -2.478e+00    -0.2673     0.9296     2.0958  4.331e+00
## beta[7]  -8.270e+00    -5.7074    -4.3306    -2.9416 -3.169e-01
```

```
## beta[8]  -5.610e+00    -3.4684    -2.3007    -1.1580  9.259e-01
## beta[9]  -1.656e+00    -0.6987    -0.1943     0.3057  1.333e+00
## beta[10] -9.399e+00    -3.0262     0.3121     3.5945  1.001e+01
## beta[11] -1.345e+01     0.7508     7.5920    14.4173  2.794e+01
## beta[12] -2.538e+00    -0.7296     0.1781     1.0496  2.831e+00
## beta[13]  3.415e-03     3.6423     5.4971     7.3521  1.120e+01
## beta[14] -2.057e+02   -76.9294    -5.9025    60.0666  1.892e+02
## beta[15] -1.603e+01    -5.8381    -0.8249     4.0496  1.364e+01
## beta_0   -2.161e+02   -88.7767   -20.5368    47.9667  1.824e+02
## deviance  6.522e+02   657.6817   661.4191   666.0759  6.767e+02
## sigma2    4.770e+04 65030.8427 76743.9424 90887.1611  1.279e+05
```

We see that variable 5 and variable 14 are the most negatively associated with crime rate, and variable 4 and variable 3 are the most positively associated with crime rate.

### (b)

Now we do a cross validation of the model.

```
#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
```

```r
  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
```

```r
chains_2b = as.mcmc(fit_JAGS_flat_pred)

##plot the predictive values
```

```r
result = summary(chains_2b)
q = result$quantiles
dtfr = as.data.frame(cbind(result$quantiles))

##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]  706.1451  915.2931 1698.4557  196.8478  852.9350 2093.4674  791.0142
##  [8]  277.2431 1291.0676  649.4401  770.7027 1267.6458  761.8144  577.7379
## [15]  736.2551  984.1269 1393.9211  536.1425  501.5114  877.0196  582.2384
## [22]  550.0983 1328.3639  586.9215
```

### (c)

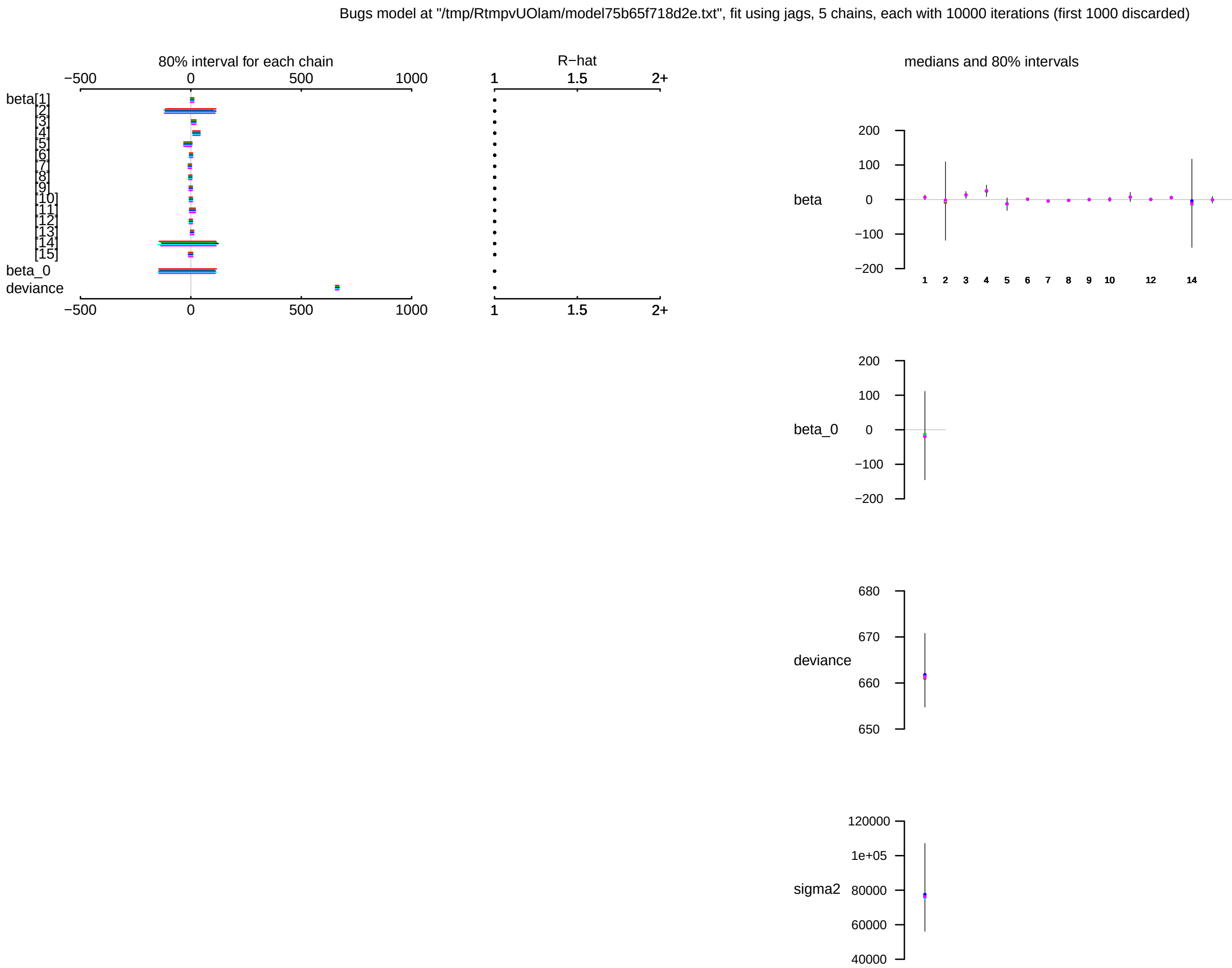
Finally, we try the spike-and-slab priors for the coefficients.

```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,
```

```r
                            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
```

```r
plot(fit_JAGS_SpikeSlab)
```



```r
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]       6.2383 5.462e+00   0.07725        0.07726
## beta[2]      -4.3728 8.926e+01   1.26235        1.25098
## beta[3]      13.2270 8.197e+00   0.11593        0.11217
## beta[4]      25.3527 1.321e+01   0.18681        0.18673
## beta[5]     -13.2961 1.454e+01   0.20566        0.20555
## beta[6]       0.9147 1.743e+00   0.02465        0.02489
## beta[7]      -4.3233 2.045e+00   0.02893        0.02858
## beta[8]      -2.3680 1.709e+00   0.02417        0.02399
## beta[9]      -0.1958 7.348e-01   0.01039        0.01030
## beta[10]      0.2830 5.057e+00   0.07152        0.07203
## beta[11]      7.3472 1.059e+01   0.14977        0.15115
## beta[12]      0.1739 1.335e+00   0.01887        0.01909
## beta[13]      5.5603 2.843e+00   0.04021        0.03980
## beta[14]    -16.5267 1.821e+02   2.57467        2.57517
## beta[15]     -0.7856 7.599e+00   0.10747        0.10577
## beta_0      -22.7384 2.133e+02   3.01653        3.01771
## deviance    662.2243 6.596e+00   0.09329        0.09325
## sigma2    79338.4890 2.098e+04 296.63763      289.35988
##
## 2. Quantiles for each variable:
##
##                 2.5%        25%        50%        75%      97.5%
## beta[1]      -4.4829     2.6292     6.2242     9.8433  1.693e+01
## beta[2]    -182.2113   -64.4574    -3.6020    55.0406  1.698e+02
## beta[3]      -3.3281     7.8743    13.2872    18.8235  2.892e+01
## beta[4]      -0.4119    16.8592    25.2690    34.0979  5.157e+01
## beta[5]     -41.9459   -22.8671   -13.3008    -3.8208  1.572e+01
## beta[6]      -2.5564    -0.2638     0.8980     2.0646  4.344e+00
## beta[7]      -8.2926    -5.7030    -4.3653    -2.9834 -1.588e-01
## beta[8]      -5.7175    -3.4823    -2.3880    -1.2231  9.482e-01
## beta[9]      -1.6281    -0.6817    -0.1920     0.2915  1.261e+00
## beta[10]     -9.7432    -2.9855     0.3046     3.6489  1.009e+01
## beta[11]    -12.9831     0.3187     7.0303    14.4697  2.868e+01
## beta[12]     -2.4528    -0.7205     0.1670     1.0873  2.754e+00
## beta[13]      0.1438     3.6775     5.5322     7.4340  1.129e+01
## beta[14]   -209.5489   -81.9757   -11.1738    55.0662  1.859e+02
## beta[15]    -15.4851    -5.7256    -0.8279     4.2465  1.428e+01
## beta_0     -215.0312   -85.1678   -17.1526    51.0012  1.878e+02
## deviance    652.1888   657.5328   661.4964   665.9663  6.775e+02
## sigma2    48258.1193 64844.0921 76273.3661 89969.1602  1.296e+05
```

```r
##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)
```

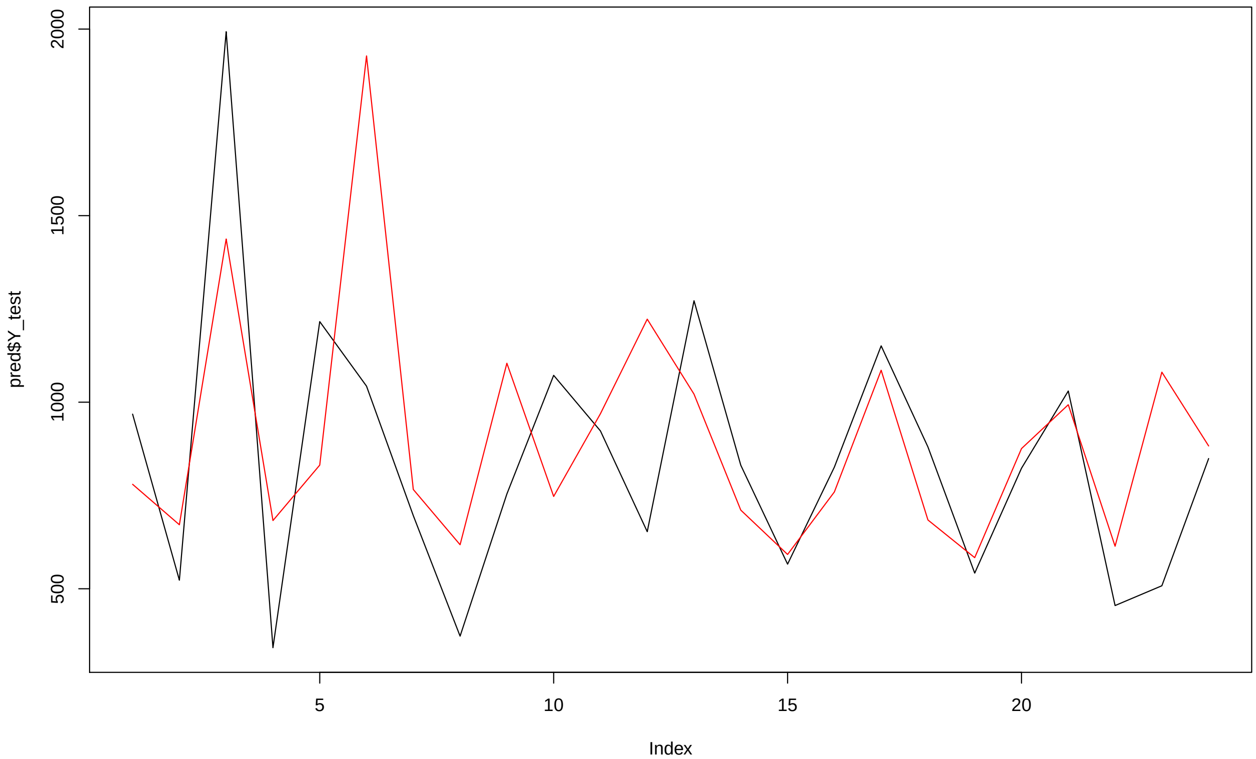
```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
```

```
## 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  779.9714
##  [2,]  523  671.5652
##  [3,] 1993 1437.3804
##  [4,]  342  682.8186
##  [5,] 1216  831.4491
##  [6,] 1043 1928.0922
##  [7,]  696  766.0820
##  [8,]  373  617.9957
##  [9,]  754 1104.4138
## [10,] 1072  747.4429
## [11,]  923  969.2428
## [12,]  653 1222.5319
## [13,] 1272 1021.6338
## [14,]  831  710.7412
## [15,]  566  591.9703
## [16,]  826  759.4924
## [17,] 1151 1085.6612
## [18,]  880  684.3600
## [19,]  542  583.3606
## [20,]  823  875.6829
## [21,] 1030  993.2718
## [22,]  455  613.9846
## [23,]  508 1080.5190
## [24,]  849  882.7556
```

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




## Problem 3

### (a)

Malaria is spread by mosquitos, so the effect of the use of bed-nets depends on the prevalence of mosquitos, which differes by village.

### (b)

We investiage whether we should allow the intercept or the slope to be random.

```
# Load dataset
gb <- gambia

# Create village number as a variable
v_loc <- unique(gb[,"x"])
v <- match(gb[,"x"],v_loc)
gb$v <- v

# Find the villages that have both netuse and nonetuse
v.netuse0 <- unique(gb$v[gb$netuse==0])
v.netuse1 <- unique(gb$v[gb$netuse==1])
v.netuse01 <- intersect(v.netuse0, v.netuse1)

# Find the pos rate in each village for netuse and nonetuse
gb.v.netuse <- aggregate(
    list(pos=gb$pos, popu=1),
    by = list(v=gb$v, netuse=gb$netuse),
```

```r
    FUN = sum
)
gb.v.netuse$pos.rate <- gb.v.netuse$pos / gb.v.netuse$popu

# Put the two lines for each village in one line
gb.v <- merge(
    gb.v.netuse[gb.v.netuse$netuse==0, c("v", "pos", "popu", "pos.rate")],
    gb.v.netuse[gb.v.netuse$netuse==1, c("v", "pos", "popu", "pos.rate")],
    by = "v",
    all = TRUE
)
names(gb.v) <- c("v",
                 "pos.nonet", "popu.nonet", "rate.nonet",
                 "pos.net", "popu.net", "rate.net"
                 )
gb.v$effect <- gb.v$rate.net - gb.v$rate.nonet

# Show the dataframes
print(str(gb))
```

```
## 'data.frame':    2035 obs. of  9 variables:
##  $ x      : num  349631 349631 349631 349631 349631 ...
##  $ y      : num  1458055 1458055 1458055 1458055 1458055 ...
##  $ pos    : num  1 0 0 1 0 1 1 0 0 1 ...
##  $ age    : num  1783 404 452 566 598 ...
##  $ netuse : num  0 1 1 1 1 1 1 1 1 0 ...
##  $ treated: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ green  : num  40.9 40.9 40.9 40.9 40.9 ...
##  $ phc    : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ v      : int  1 1 1 1 1 1 1 1 1 1 ...
## NULL
```

```r
print(str(gb.v))
```

```
## 'data.frame':    65 obs. of  8 variables:
##  $ v         : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ pos.nonet : num  5 6 NA 7 NA 2 0 0 6 7 ...
##  $ popu.nonet: num  6 17 NA 20 NA 7 1 6 57 25 ...
##  $ rate.nonet: num  0.833 0.353 NA 0.35 NA ...
##  $ pos.net   : num  12 13 7 1 10 5 24 7 NA 1 ...
##  $ popu.net  : num  27 46 17 4 26 11 37 50 NA 1 ...
##  $ rate.net  : num  0.444 0.283 0.412 0.25 0.385 ...
##  $ effect    : num  -0.3889 -0.0703 NA -0.1 NA ...
## NULL
```

```r
# Linear regression MCMC by JAGS
# Fixed intercept and slope
linear.model.JAGS <- function(){
    for(i in 1:n){
        y[i] ~ dbern(p[i])
        p[i] <- exp(logitp[i]) / (1 + exp(logitp[i]))
        logitp[i] <- alpha + beta * x[i]
    }
    alpha ~ dnorm(mu.alpha, tausq.alpha)
    beta ~ dnorm(mu.beta, tausq.beta)
```

```
}

# Random intercept
linear.model.JAGS.randalpha <- function(){
    for(i in 1:n){
        y[i] ~ dbern(p[i])
        p[i] <- exp(logitp[i]) / (1 + exp(logitp[i]))
        logitp[i] <- alpha[v[i]] + beta * x[i]
    }
    for(j in 1:m){
        alpha[j] ~ dnorm(mu.alpha, tausq.alpha)
    }
    beta ~ dnorm(mu.beta, tausq.beta)
}

# Random slope
linear.model.JAGS.randbeta <- function(){
    for(i in 1:n){
        y[i] ~ dbern(p[i])
        p[i] <- exp(logitp[i]) / (1 + exp(logitp[i]))
        logitp[i] <- alpha + beta[v[i]] * x[i]
    }
    for(j in 1:m){
        beta[j] ~ dnorm(mu.beta, tausq.beta)
    }
    alpha ~ dnorm(mu.alpha, tausq.alpha)
}

# Random intercept and slope
# Data: y, x, v, n = length(y), m = length(v)
linear.model.JAGS.randalphabeta <- function(){
    for(i in 1:n){
        y[i] ~ dbern(p[i])
        p[i] <- exp(logitp[i]) / (1 + exp(logitp[i]))
        logitp[i] <- alpha[v[i]] + beta[v[i]] * x[i]
    }
    for(j in 1:m){
        alpha[j] ~ dnorm(mu.alpha, tausq.alpha)
        beta[j] ~ dnorm(mu.beta, tausq.beta)
    }
}

dat.JAGS <- list(y = gb$pos,
                x = gb$netuse,
                v = gb$v,
                n = nrow(gb),
                m = length(unique(gb$v)),
                mu.alpha = 0,
                tausq.alpha = 1e-3,
                mu.beta = 0,
                tausq.beta = 1e-3
                )
para.JAGS <- c("alpha", "beta")
```

```r
fit.JAGS.randalphabeta <- jags(data = dat.JAGS,
                 parameters.to.save = para.JAGS,
                 ## inits = inits.JAGS
                 n.chains = 1,
                 n.iter = 1e4,
                 n.burnin = 1e3,
                 model.file = linear.model.JAGS.randalphabeta
                 )
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 2035
##    Unobserved stochastic nodes: 130
##    Total graph size: 8811
##
## Initializing model
```

```r
fit.JAGS <- jags(data = dat.JAGS,
                 parameters.to.save = para.JAGS,
                 ## inits = inits.JAGS
                 n.chains = 1,
                 n.iter = 1e4,
                 n.burnin = 1e3,
                 model.file = linear.model.JAGS
                 )
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 2035
##    Unobserved stochastic nodes: 2
##    Total graph size: 6122
##
## Initializing model
```

```r
fit.JAGS.randalpha <- jags(data = dat.JAGS,
                 parameters.to.save = para.JAGS,
                 ## inits = inits.JAGS
                 n.chains = 1,
                 n.iter = 1e4,
                 n.burnin = 1e3,
                 model.file = linear.model.JAGS.randalpha
                 )
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 2035
##    Unobserved stochastic nodes: 66
##    Total graph size: 8642
##
```

```
## Initializing model
```

```r
fit.JAGS.randbeta <- jags(data = dat.JAGS,
                 parameters.to.save = para.JAGS,
                 ## inits = inits.JAGS
                 n.chains = 1,
                 n.iter = 1e4,
                 n.burnin = 1e3,
                 model.file = linear.model.JAGS.randbeta
                 )
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 2035
##    Unobserved stochastic nodes: 66
##    Total graph size: 8747
##
## Initializing model
```

```r
## Show DIC and pD for model comparison
## Fixed intercept and slope
print(c(fit.JAGS$BUGSoutput$DIC,
        fit.JAGS$BUGSoutput$pD))
```

```
## [1] 2599.912912    2.256292
```

```r
## Random intercept
print(c(fit.JAGS.randalpha$BUGSoutput$DIC,
        fit.JAGS.randalpha$BUGSoutput$pD))
```

```
## [1] 2372.78490   71.75505
```

```r
## Random slope
print(c(fit.JAGS.randbeta$BUGSoutput$DIC,
        fit.JAGS.randbeta$BUGSoutput$pD))
```

```
## [1] 2461.53060   66.01548
```

```r
## Random intercept and slope
print(c(fit.JAGS.randalphabeta$BUGSoutput$DIC,
        fit.JAGS.randalphabeta$BUGSoutput$pD))
```

```
## [1] 2380.3987  106.1909
```

We see that the model with a random intercept and a fixed slope has the lowest DIC and thus is the most informative and economic model.

### (c)

We now find the villages with the least/largest intercept/effect.

```r
# Extract values of alpha and beta from JAGS output
alpha.result <- fit.JAGS.randalphabeta$BUGSoutput$mean$alpha
beta.result <- fit.JAGS.randalphabeta$BUGSoutput$mean$beta
alpha.result[-v.netuse01] <- NA
beta.result[-v.netuse01] <- NA
```

```r
rate.nonet.result <- round(inv.logit(alpha.result), 3)
rate.net.result <- round(inv.logit(alpha.result + beta.result),3)
effect.result <- rate.net.result - rate.nonet.result

print("Observed extremes")
```

```
## [1] "Observed extremes"
```

```r
print(c(which.min(rate.nonet.result), min(rate.nonet.result, na.rm = TRUE)))
```

```
## [1] 7 0
```

```r
print(c(which.max(rate.nonet.result), max(rate.nonet.result, na.rm = TRUE)))
```

```
## [1] 13  1
```

```r
print(c(which.min(effect.result), min(effect.result, na.rm = TRUE)))
```

```
## [1] 13.000 -0.606
```

```r
print(c(which.max(effect.result), max(effect.result, na.rm = TRUE)))
```

```
## [1] 10.000  0.721
```

```r
print("Regression extremes")
```

```
## [1] "Regression extremes"
```

```r
gb.v[which.min(gb.v$rate.nonet), c("v", "rate.nonet")]
```

```
##   v rate.nonet
## 7 7          0
```

```r
gb.v[which.max(gb.v$rate.nonet), c("v", "rate.nonet")]
```

```
##     v rate.nonet
## 13 13          1
```

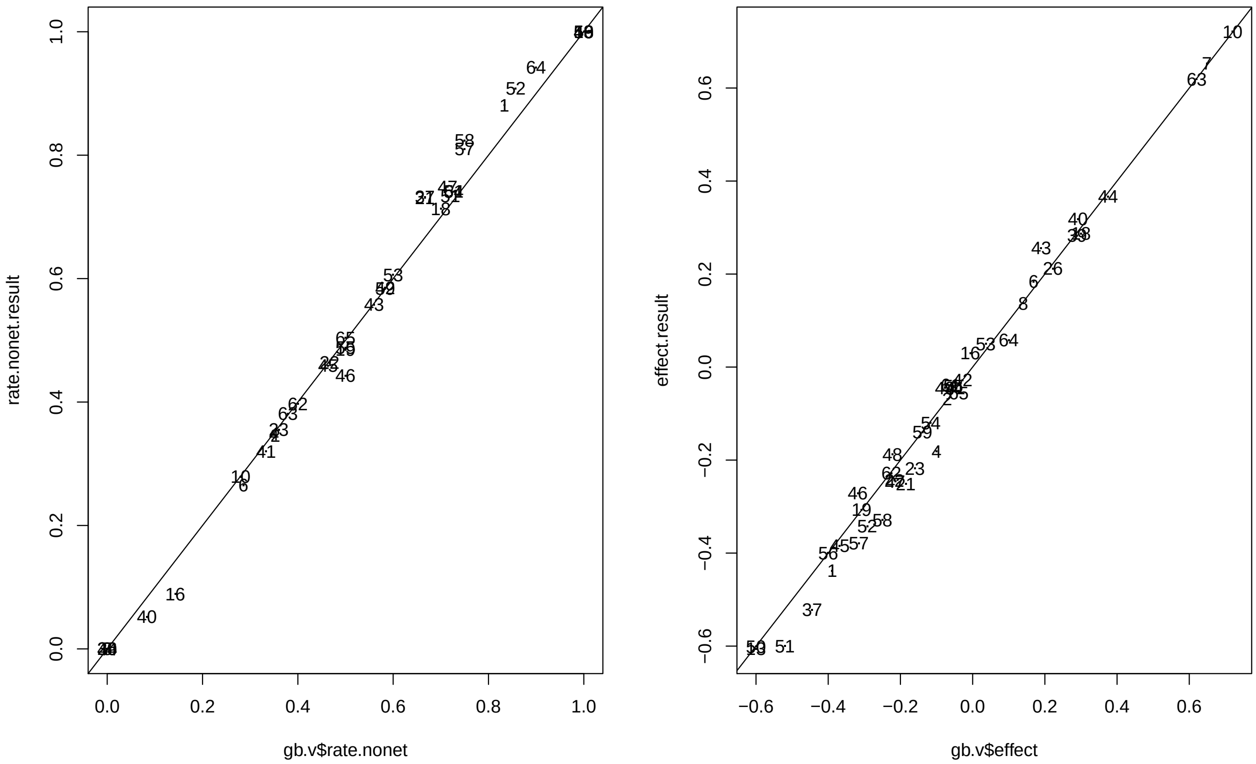
```r
gb.v[which.min(gb.v$effect), c("v", "effect")]
```

```
##     v effect
## 13 13   -0.6
```

```r
gb.v[which.max(gb.v$effect), c("v", "effect")]
```

```
##     v effect
## 10 10   0.72
```

```r
# Compare the regression result with direct observation
par(mfrow=c(1,2))
v.num <- nrow(gb.v)
plot(gb.v$rate.nonet, rate.nonet.result, pch=".")
text(gb.v$rate.nonet, rate.nonet.result, c(1:v.num))
abline(0,1)
plot(gb.v$effect, effect.result, pch=".")
text(gb.v$effect, effect.result, c(1:v.num))
abline(0,1)
```



We see that the regression result and direct observation give similar results Specifically,

- Village with the smallest intercept: 7
- Village with the greatest intercept: 13
- Village with the most decrease of infection by using bednets: 13 (direct observation, effect = -0.605), 50 (regression, effect = -0.6)
- Village with the most decrease of infection by using bednets: 10

### (d)

Finally, we explore the influcence of the prior.

```
# Larger precision
dat.JAGS.largeTausq <- list(y = gb$pos,
                x = gb$netuse,
                v = gb$v,
                n = nrow(gb),
                m = length(unique(gb$v)),
                mu.alpha = 0,
                tausq.alpha = 1e0,
                mu.beta = 0,
                tausq.beta = 1e0
                )
fit.JAGS.randalphabeta.largeTausq <- jags(data = dat.JAGS.largeTausq,
                 parameters.to.save = para.JAGS,
                 ## inits = inits.JAGS
                 n.chains = 1,
                 n.iter = 1e4,
                 n.burnin = 1e3,
                 model.file = linear.model.JAGS.randalphabeta
```

```
                 )
```

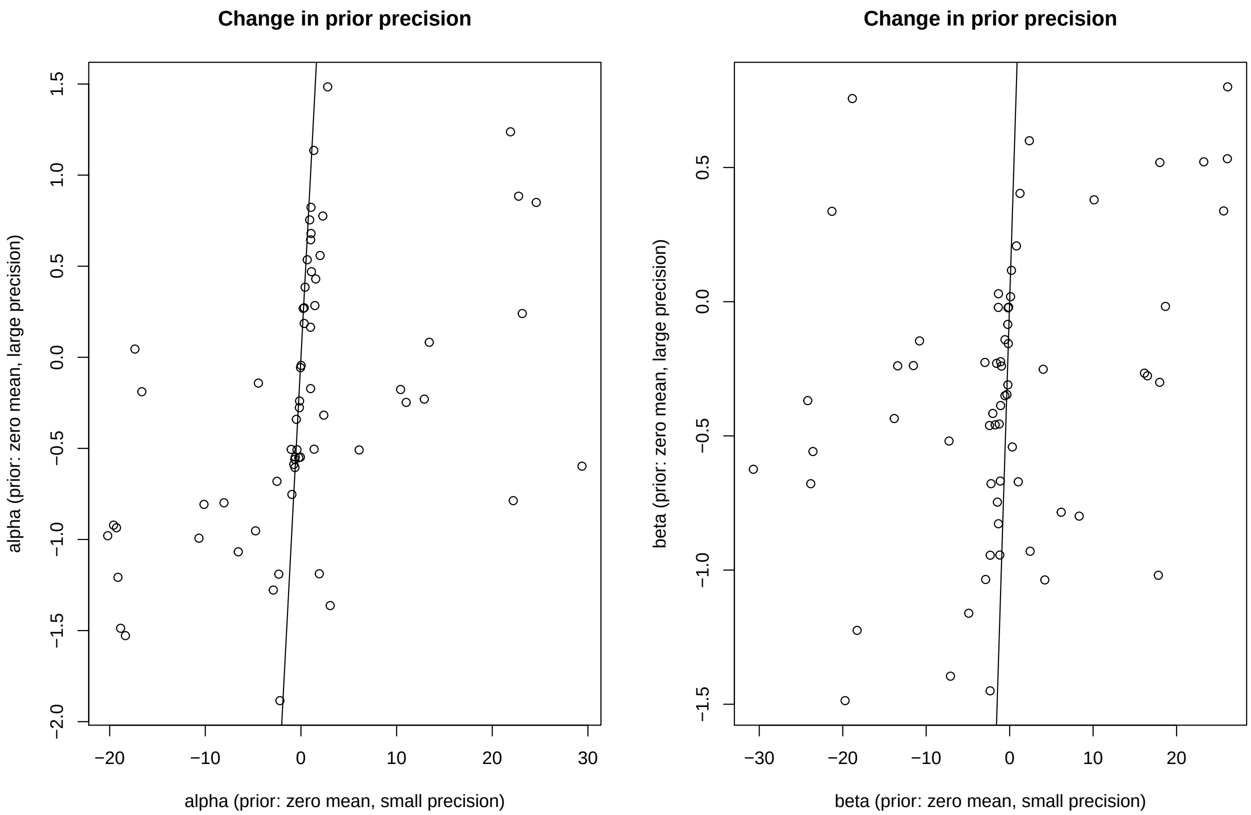
```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 2035
##    Unobserved stochastic nodes: 130
##    Total graph size: 8811
##
## Initializing model
```

```
# Nonzero means
dat.JAGS.offMu <- list(y = gb$pos,
                x = gb$netuse,
                v = gb$v,
                n = nrow(gb),
                m = length(unique(gb$v)),
                mu.alpha = logit(0.5),
                tausq.alpha = 1e-3,
                mu.beta = logit(0.4) - logit(0.6),
                tausq.beta = 1e-3
                )
fit.JAGS.randalphabeta.offMu <- jags(data = dat.JAGS.offMu,
                 parameters.to.save = para.JAGS,
                 ## inits = inits.JAGS
                 n.chains = 1,
                 n.iter = 1e4,
                 n.burnin = 1e3,
                 model.file = linear.model.JAGS.randalphabeta
                 )
```

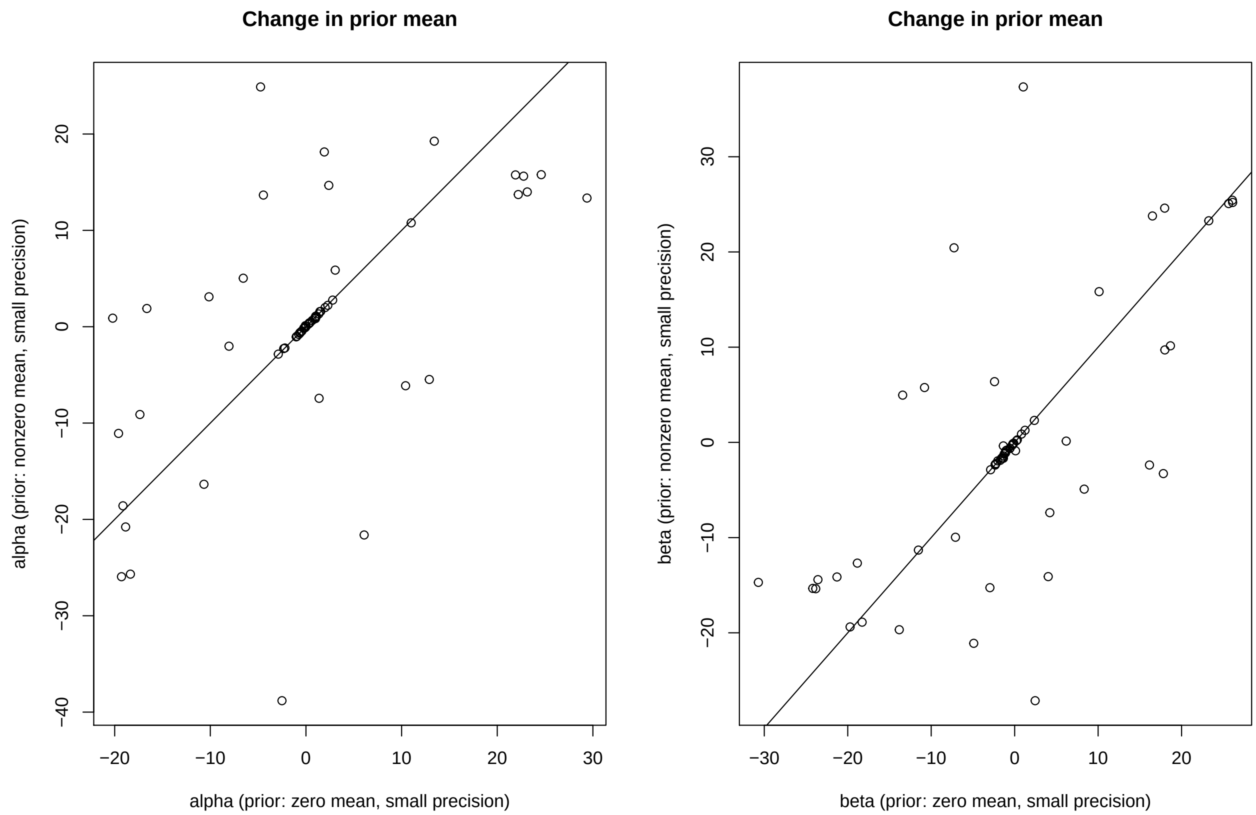
```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 2035
##    Unobserved stochastic nodes: 130
##    Total graph size: 8811
##
## Initializing model
```

```
par(mfrow=c(1,2))
plot(
    fit.JAGS.randalphabeta$BUGSoutput$mean$alpha,
    fit.JAGS.randalphabeta.largeTausq$BUGSoutput$mean$alpha,
    main = "Change in prior precision",
    xlab = "alpha (prior: zero mean, small precision)",
    ylab = "alpha (prior: zero mean, large precision)"
)
abline(0,1)
plot(
    fit.JAGS.randalphabeta$BUGSoutput$mean$beta,
    fit.JAGS.randalphabeta.largeTausq$BUGSoutput$mean$beta,
    main = "Change in prior precision",
```

```r
    xlab = "beta (prior: zero mean, small precision)",
    ylab = "beta (prior: zero mean, large precision)"
)
abline(0,1)
```




```r
plot(
    fit.JAGS.randalphabeta$BUGSoutput$mean$alpha,
    fit.JAGS.randalphabeta.offMu$BUGSoutput$mean$alpha,
    main = "Change in prior mean",
    xlab = "alpha (prior: zero mean, small precision)",
    ylab = "alpha (prior: nonzero mean, small precision)"
)
abline(0,1)
plot(
    fit.JAGS.randalphabeta$BUGSoutput$mean$beta,
    fit.JAGS.randalphabeta.offMu$BUGSoutput$mean$beta,
    main = "Change in prior mean",
    xlab = "beta (prior: zero mean, small precision)",
    ylab = "beta (prior: nonzero mean, small precision)"
)
abline(0,1)
```




We see that when the prior mean is still zero but the precision is large, the posterior means are closer to the prior mean (which is equal to 1) and therefore has a magnitude less than the results given by a flatter prior. When we let the prior precision to be small but the mean to be nonzero, the posterior means are mostly in the same scale as the results given by the zero-mean prior, although the means are now more spread out. Thus in our case, an overconfident prior is more lethal than a moderate prior, when we do not have good prior knowledge about the means.