Bayesian linear regression

Jingchen (Monika) Hu

MATH 347 Bayesian Statistics

Installing the necessary packages

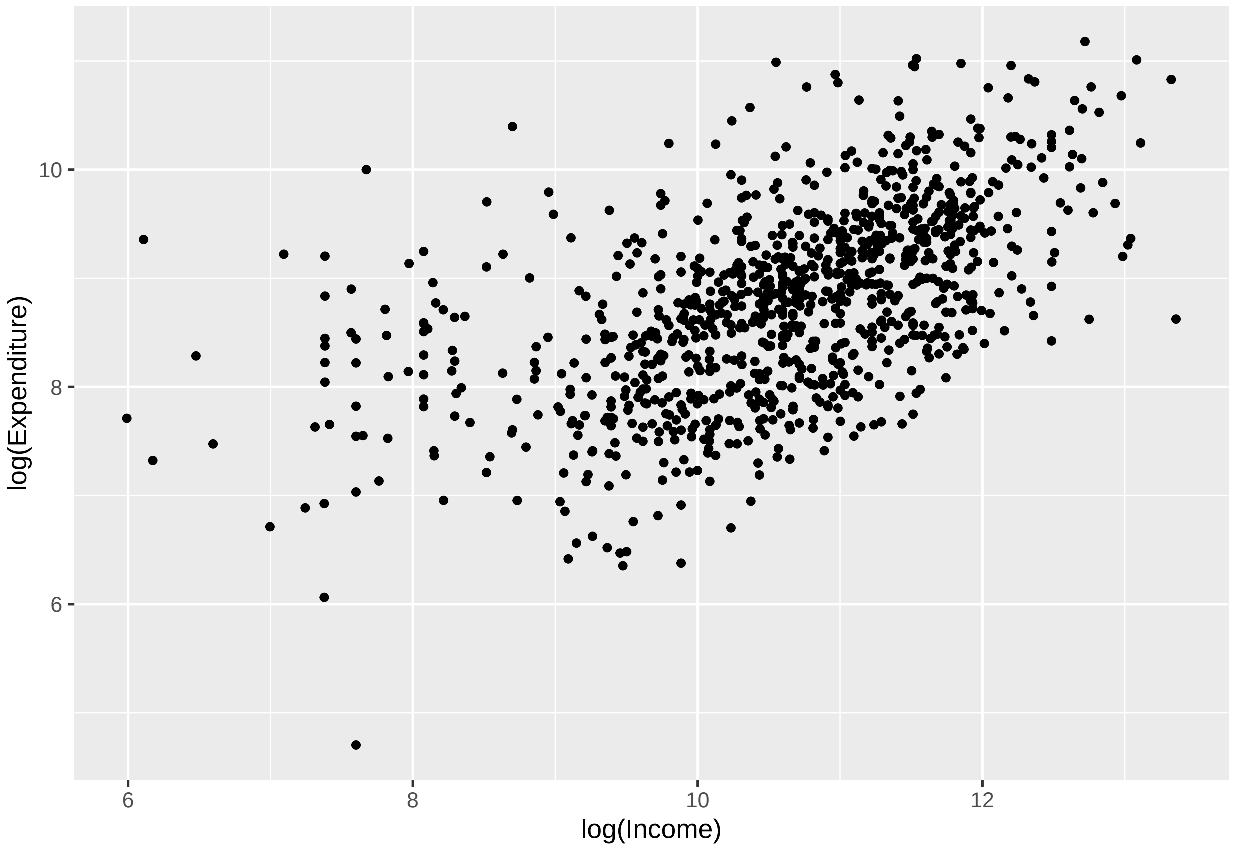
```
install.packages("devtools")
require(devtools)
devtools::install_github("bayesball/ProbBayes")

require(ggplot2)
require(gridExtra)
require(ProbBayes)
require(tidyverse)
crcblue <- "#2905a1"
```

Introduction: Adding a continuous predictor variable

The simple linear regression model

```
CEData <- read.csv("CEsample.csv", header = T, sep = ",")
g1 <- ggplot(CEData, aes(x = log_TotalIncome, y = log_TotalExp)) +
  geom_point(size=1) +
  labs(x = "log(Income)", y = "log(Expenditure)") +
  theme_grey(base_size = 10, base_family = "")
g1
```



The CE sample

A simple linear regression for the CE sample

MCMC simulation by JAGS for the SLR model

JAGS script for the SLR model

```
modelString <-"
model {
## sampling
for (i in 1:N){
y[i] ~ dnorm(beta0 + beta1*x[i], invsigma2)
}

## priors
beta0 ~ dnorm(mu0, g0)
beta1 ~ dnorm(mu1, g1)
invsigma2 ~ dgamma(a, b)
sigma <- sqrt(pow(invsigma2, -1))
}
"
```

- Pass the data and hyperparameter values to JAGS:

```
y <- as.vector(CEData$log_TotalExp)
x <- as.vector(CEData$log_TotalIncome)
N <- length(y)
the_data <- list("y" = y, "x" = x, "N" = N,
                 "mu0" = 0, "g0" = 0.0001,
                 "mu1" = 0, "g1" = 0.0001,
                 "a" = 1, "b" = 1)

initsfunction <- function(chain){
  .RNG.seed <- c(1,2)[chain]
  .RNG.name <- c("base::Super-Duper",
                 "base::Wichmann-Hill")[chain]
  return(list(.RNG.seed=.RNG.seed,
              .RNG.name=.RNG.name))
}
```

- Run the JAGS code for this model:

```
posterior <- run.jags(modelString,
                      n.chains = 1,
                      data = the_data,
                      monitor = c("beta0", "beta1", "sigma"),
                      adapt = 1000,
                      burnin = 5000,
                      sample = 5000,
                      thin = 1,
                      inits = initsfunction)
```

```
## Calling the simulation...
## Welcome to JAGS 4.3.0 on Mon Nov 25 12:21:54 2019
## JAGS is free software and comes with ABSOLUTELY NO WARRANTY
## Loading module: basemod: ok
## Loading module: bugs: ok
## . . Reading data file data.txt
## . Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 994
##    Unobserved stochastic nodes: 3
##    Total graph size: 3466
## . Reading parameter file inits1.txt
## . Initializing model
## . Adaptation skipped: model is not in adaptive mode.
## . Updating 5000
## -------------------------------------------------| 5000
## ************************************************** 100%
## . . . . Updating 5000
## -------------------------------------------------| 5000
## ************************************************** 100%
## . . . . Updating 0
## . Deleting model
## .
## Note: the model did not require adaptation
## Simulation complete.  Reading coda files...
## Coda files loaded successfully
## Calculating summary statistics...
## Finished running the simulation
```

JAGS output for the SLR model

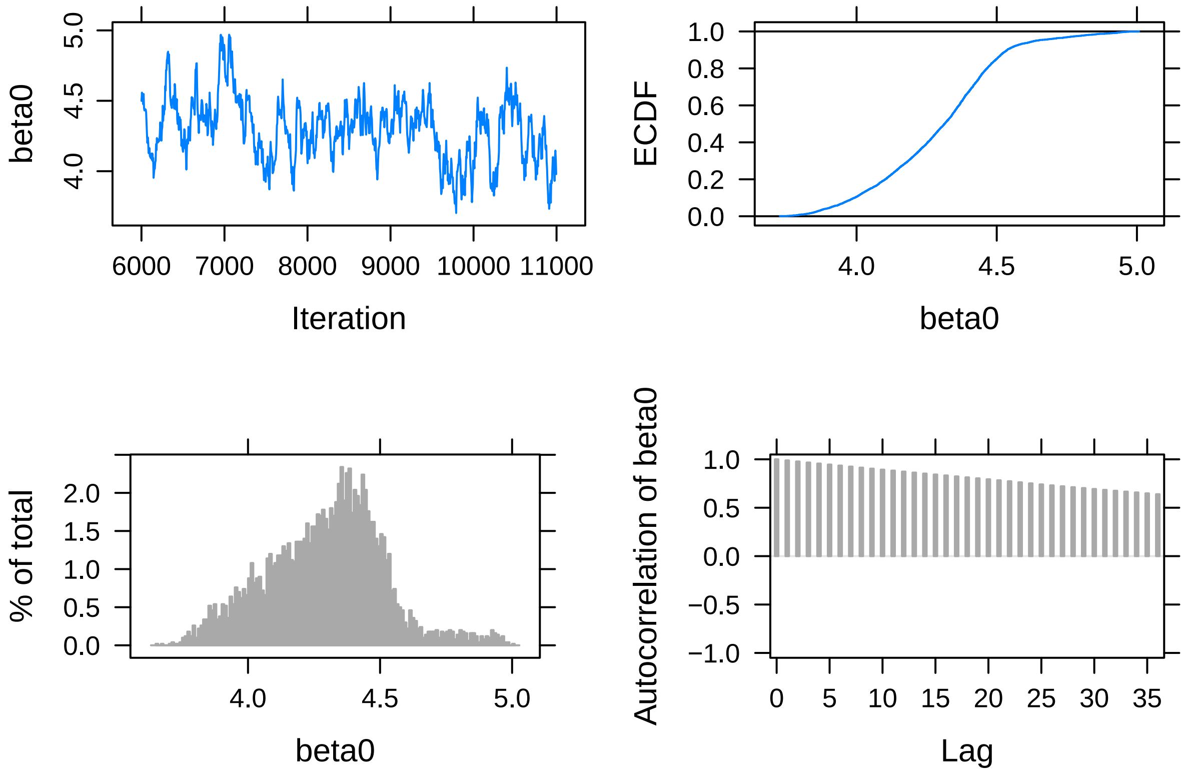
- Obtain posterior summaries of all parameters:

```r
summary(posterior)
```

```
##        Lower95    Median  Upper95      Mean         SD Mode        MCerr
## beta0 3.771420 4.3136450 4.655760 4.2947731 0.22275382   NA 0.0421194486
## beta1 0.388552 0.4218125 0.471380 0.4237176 0.02091679   NA 0.0039634093
## sigma 0.694060 0.7249680 0.757081 0.7251875 0.01624723   NA 0.0002175128
##       MC%ofSD SSeff      AC.10 psrf
## beta0    18.9    28 0.89361268   NA
## beta1    18.9    28 0.89351502   NA
## sigma     1.3  5579 0.02058294   NA
```

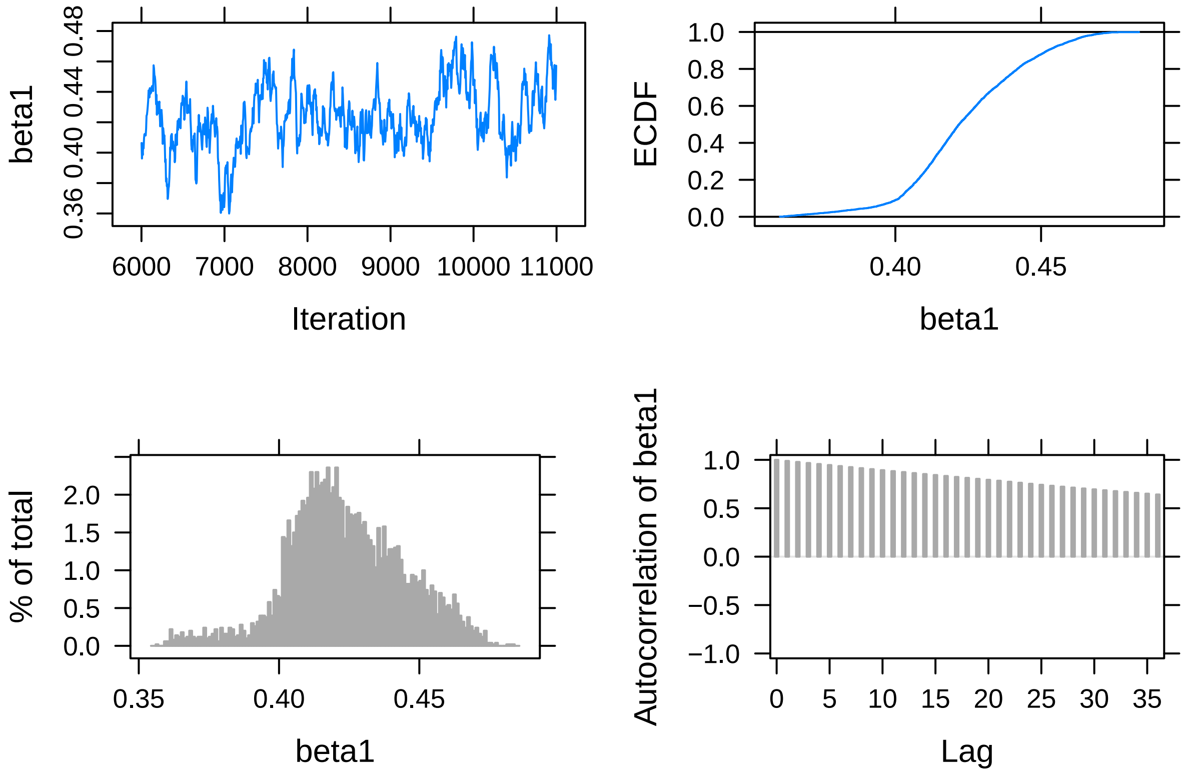
```r
plot(posterior, vars = "beta0")
```

```
## Generating plots...
```



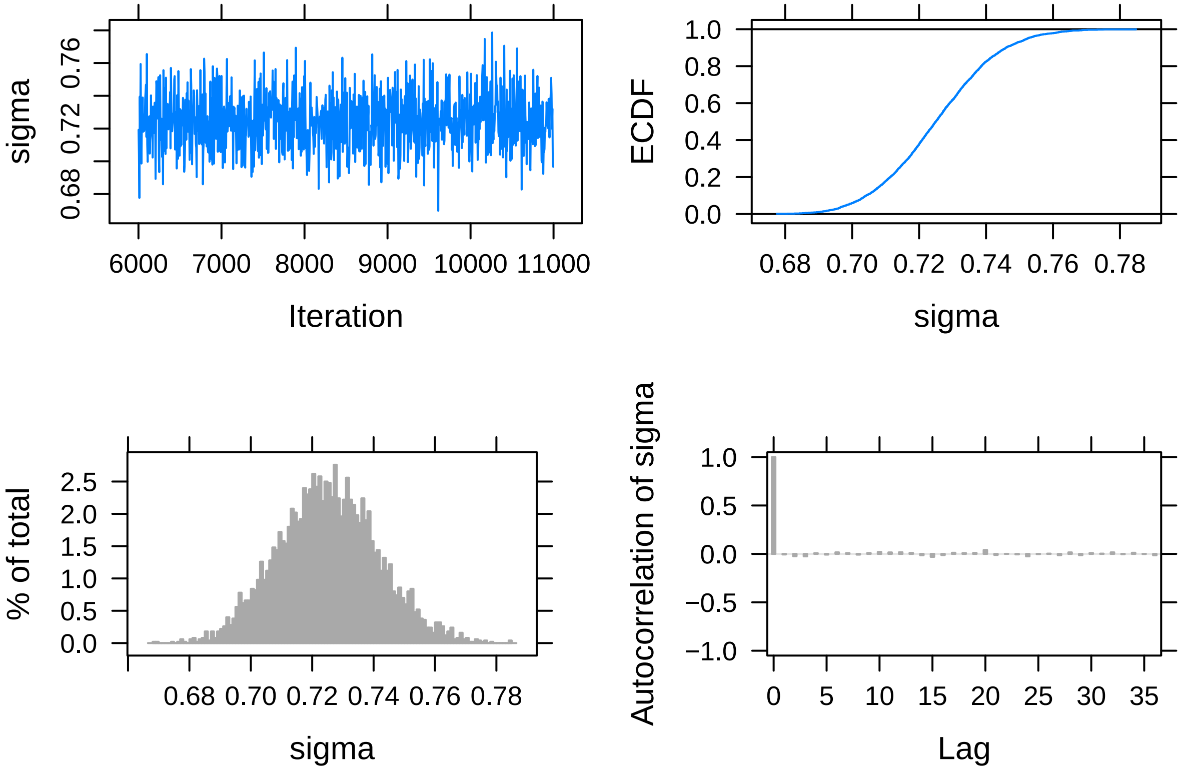
```r
plot(posterior, vars = "beta1")
```

```
## Generating plots...
```



```r
plot(posterior, vars = "sigma")
```

```
## Generating plots...
```



New JAGS script for the SLR model

Setting `thin = 50`, to get rid of the stickiness in β_0 and β_1.

```r
posterior_new <- run.jags(modelString,
                          n.chains = 1,
                          data = the_data,
                          monitor = c("beta0", "beta1", "sigma"),
                          adapt = 1000,
                          burnin = 5000,
                          sample = 5000,
                          thin = 50,
                          inits = initsfunction)
```

```
## Calling the simulation...
## Welcome to JAGS 4.3.0 on Mon Nov 25 12:21:59 2019
## JAGS is free software and comes with ABSOLUTELY NO WARRANTY
## Loading module: basemod: ok
## Loading module: bugs: ok
## . . Reading data file data.txt
## . Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 994
##    Unobserved stochastic nodes: 3
##    Total graph size: 3466
## . Reading parameter file inits1.txt
## . Initializing model
## . Adaptation skipped: model is not in adaptive mode.
## . Updating 5000
## --------------------------------------------------| 5000
## ************************************************** 100%
## . . . . Updating 250000
## --------------------------------------------------| 250000
## ************************************************** 100%
## . . . . Updating 0
## . Deleting model
## .
## Note: the model did not require adaptation
## Simulation complete.  Reading coda files...
## Coda files loaded successfully
## Calculating summary statistics...
## Finished running the simulation
```

New JAGS output for the SLR model

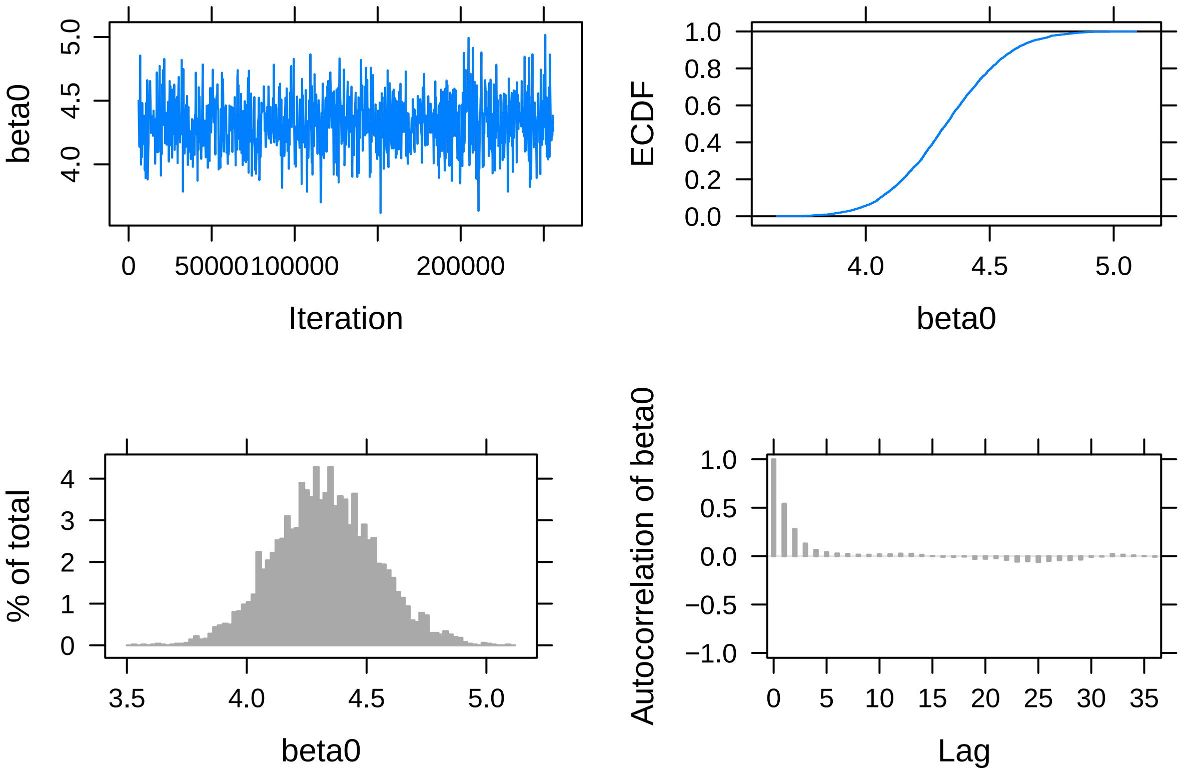
- Obtain posterior summaries of all parameters:

```r
summary(posterior_new)
```

```
##        Lower95    Median  Upper95      Mean         SD Mode        MCerr
## beta0 3.926200 4.3251800 4.753090 4.3269816 0.21092686   NA 0.0054707075
## beta1 0.381057 0.4208825 0.458968 0.4207384 0.01977946   NA 0.0004930936
## sigma 0.693623 0.7254665 0.757397 0.7255765 0.01624891   NA 0.0002297942
##       MC%ofSD SSeff      AC.500 psrf
## beta0     2.6  1487 0.018912301   NA
## beta1     2.5  1609 0.018555972   NA
## sigma     1.4  5000 0.003076245   NA
```

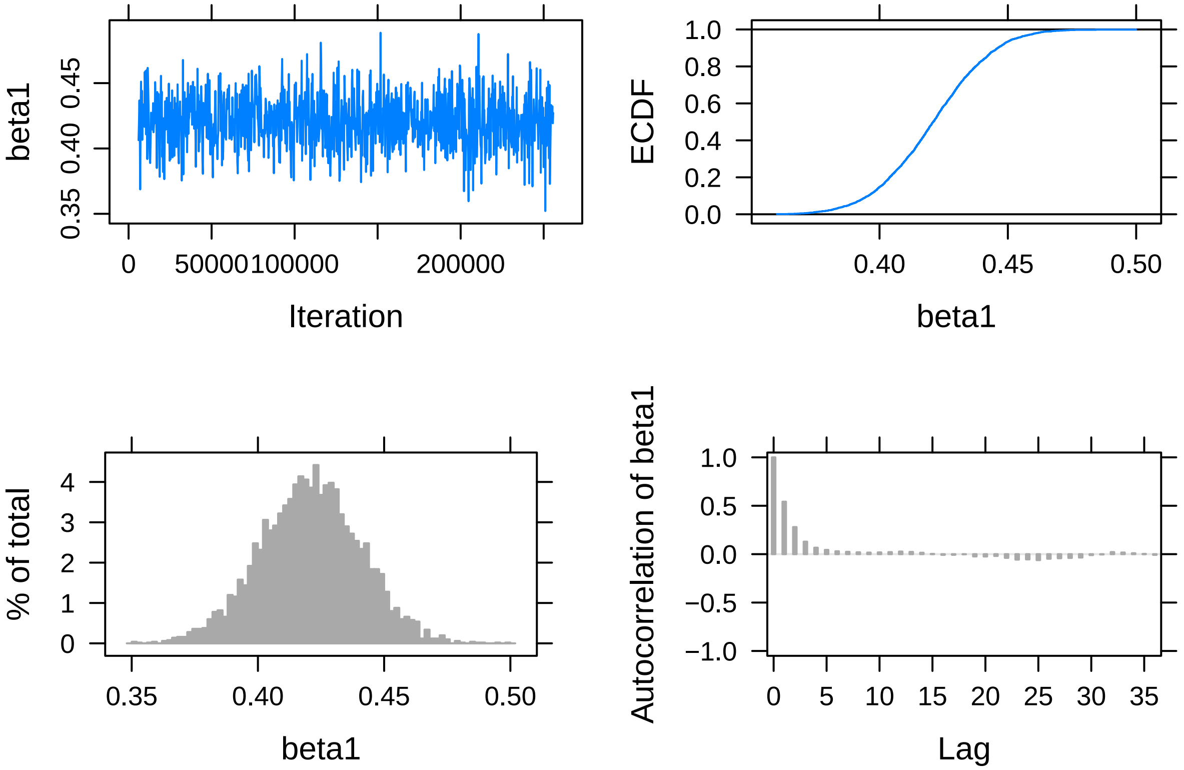
```r
plot(posterior_new, vars = "beta0")
```

```
## Generating plots...
```



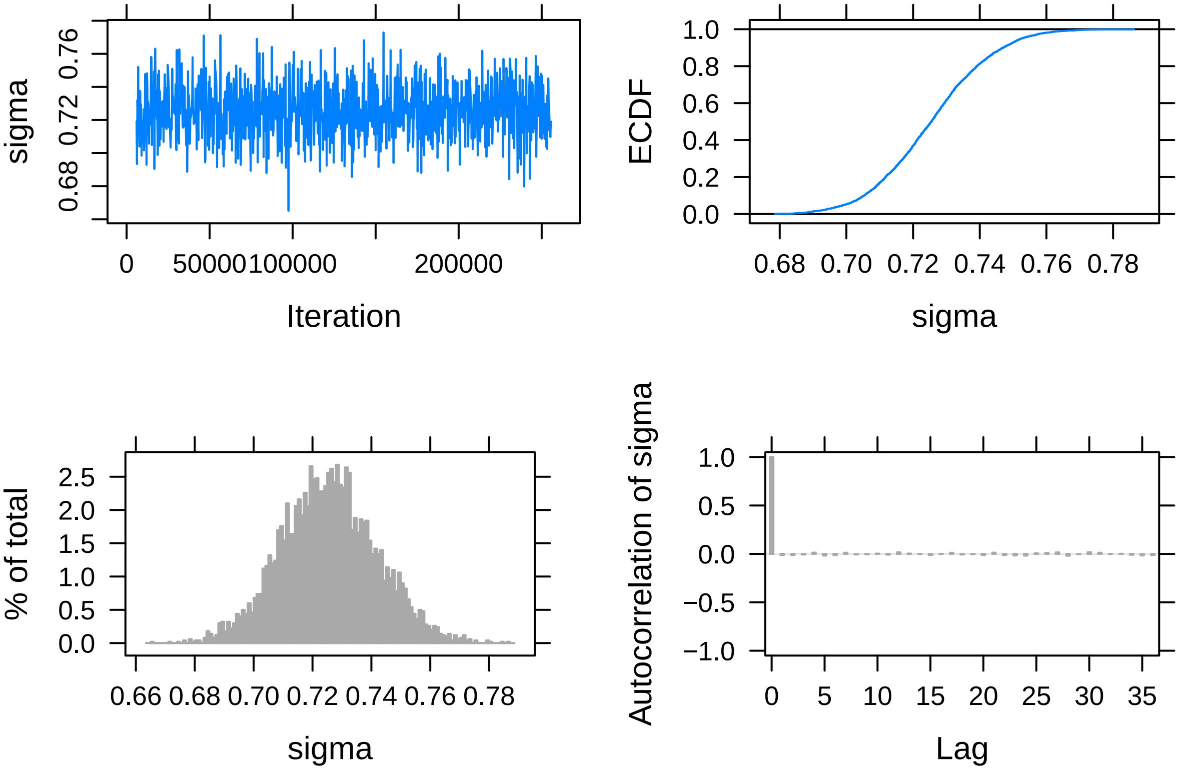
```r
plot(posterior_new, vars = "beta1")
```

```
## Generating plots...
```



```r
plot(posterior_new, vars = "sigma")
```

```
## Generating plots...
```

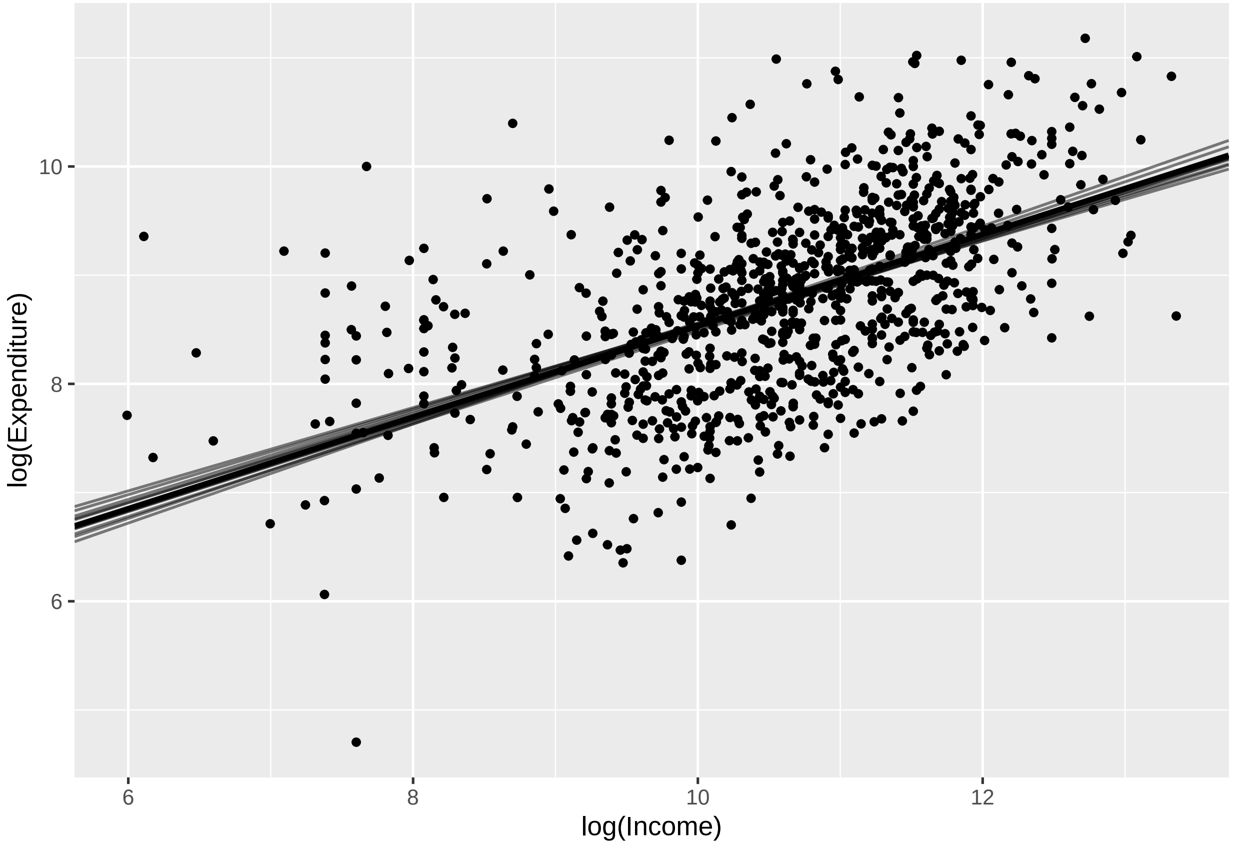


Bayesian inferences with SLR

Simulate fits from the regression model

```r
post <- as.mcmc(posterior_new)
post_means <- apply(post, 2, mean)
post <- as.data.frame(post)
```

```r
ggplot(CEData, aes(log_TotalIncome, log_TotalExp)) +
  geom_point(size=1) +
  geom_abline(data=post[1:10, ],
              aes(intercept=beta0, slope=beta1), alpha = 0.5) +
  geom_abline(intercept = post_means[1],
              slope = post_means[2], size = 1) +
  ylab("log(Expenditure)") + xlab("log(Income)") +
  theme_grey(base_size = 10, base_family = "")
```

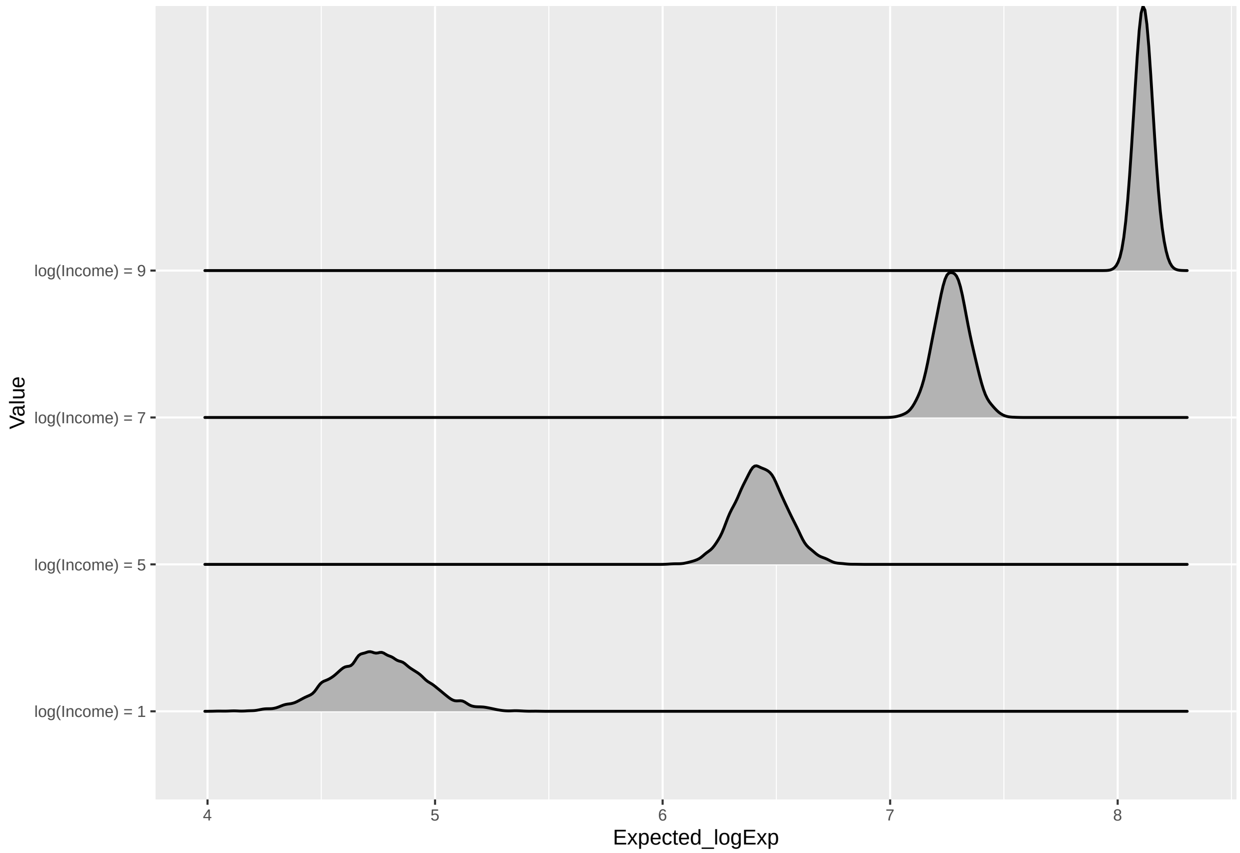


Learning about the expected response

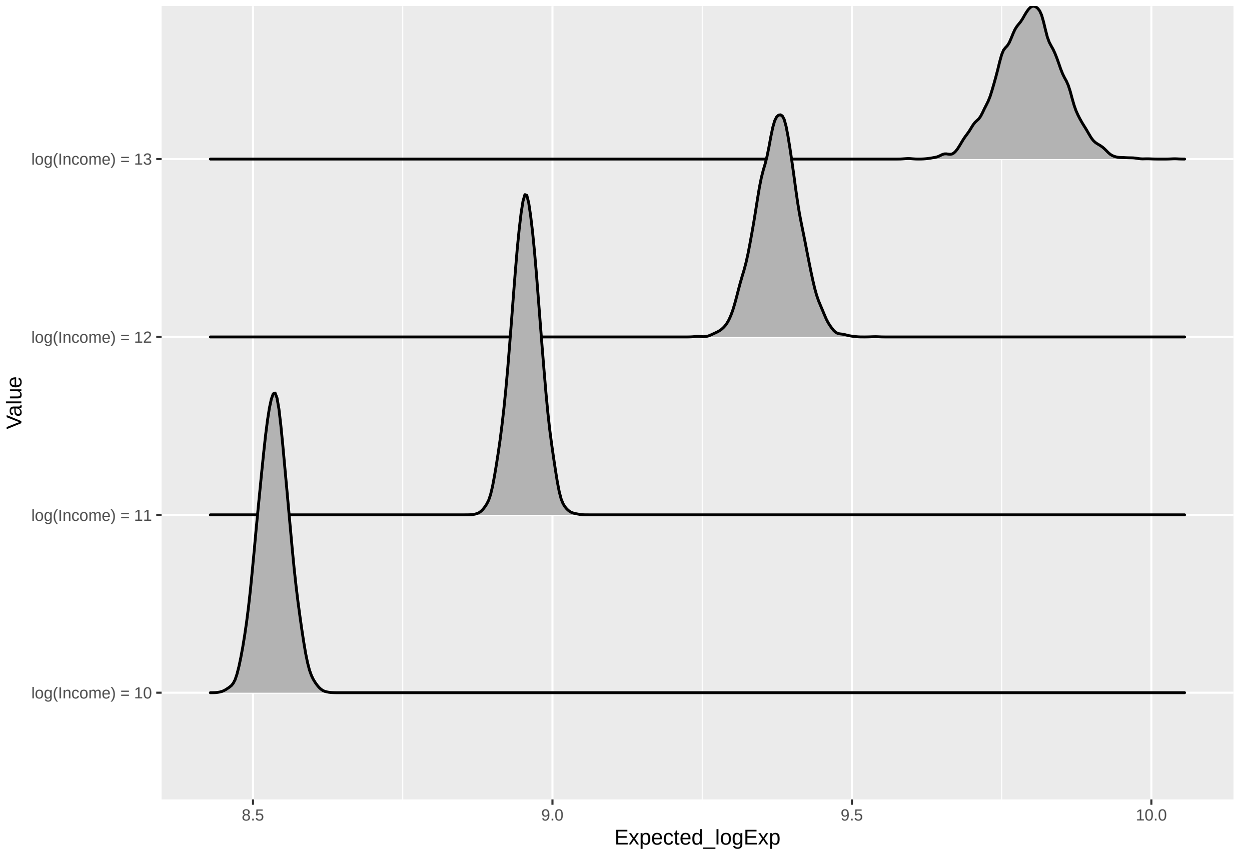
```r
post <- as.data.frame(post)
one_expected <- function(x){
  lp <- post[ , "beta0"] +  x * post[ , "beta1"]
  data.frame(Value = paste("log(Income) =", x),
             Expected_logExp = lp)
}

df <- map_df(c(1, 5, 7, 9), one_expected)
```

```r
require(ggridges)
ggplot(df, aes(x = Expected_logExp, y = Value)) +
  geom_density_ridges() +
  theme_grey(base_size = 8, base_family = "")
```



```r
df_new <- map_df(c(10, 11, 12, 13), one_expected)
ggplot(df_new, aes(x = Expected_logExp, y = Value)) +
  geom_density_ridges() +
  theme_grey(base_size = 8, base_family = "")
```



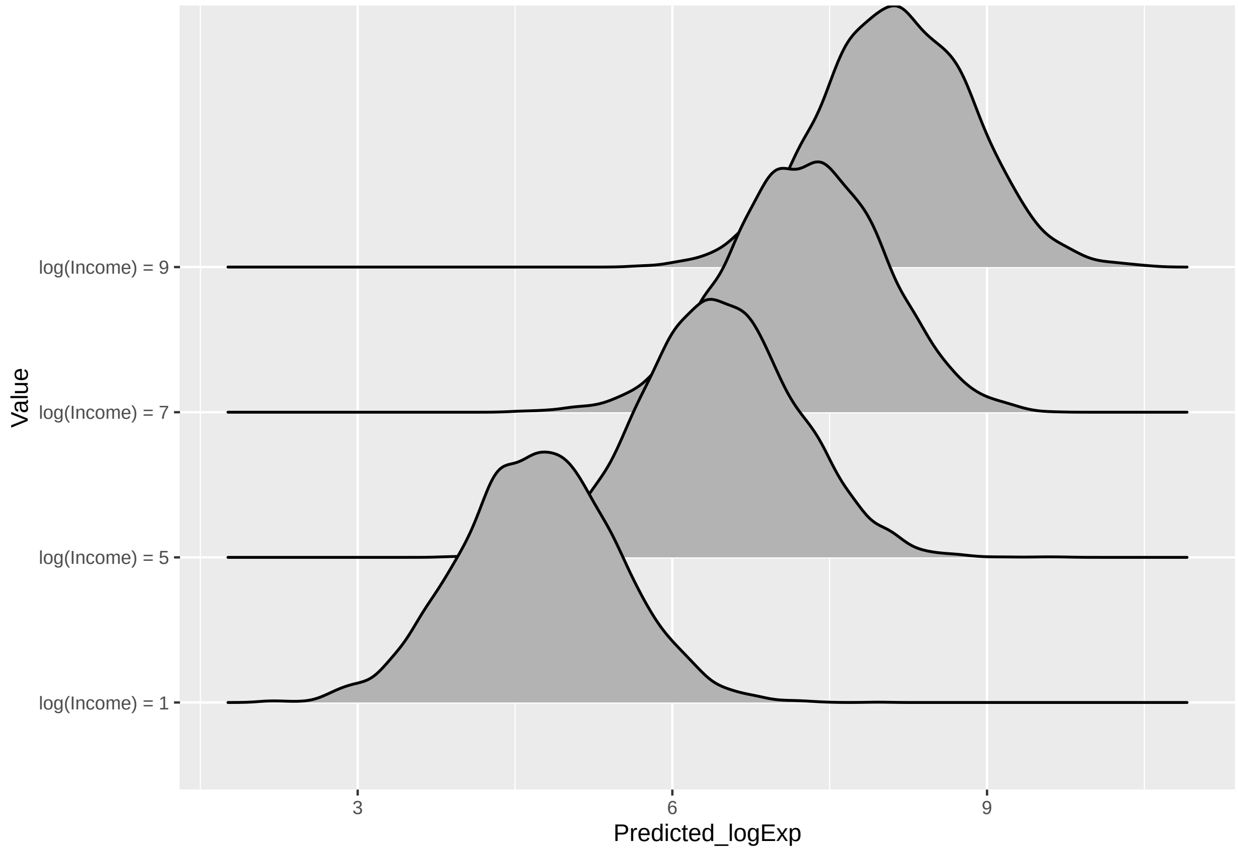
```
df %>% group_by(Value) %>%
  summarize(P05 = quantile(Expected_logExp, 0.05),
            P50 = median(Expected_logExp),
            P95 = quantile(Expected_logExp, 0.95))
```

```
## # A tibble: 4 x 4
##   Value              P05   P50   P95
##   <chr>            <dbl> <dbl> <dbl>
## 1 log(Income) = 1   4.44  4.75  5.06
## 2 log(Income) = 5   6.25  6.43  6.62
## 3 log(Income) = 7   7.15  7.27  7.40
## 4 log(Income) = 9   8.05  8.11  8.18
```

Prediction of future responses

```
one_predicted <- function(x){
  lp <- post[ , "beta0"] +  x * post[ , "beta1"]
  y <- rnorm(5000, lp, post[, "sigma"])
  data.frame(Value = paste("log(Income) =", x),
             Predicted_logExp = y)
}
df <- map_df(c(1, 5, 7, 9), one_predicted)
```

```r
require(ggridges)
ggplot(df, aes(x = Predicted_logExp, y = Value)) +
  geom_density_ridges() +
  theme_grey(base_size = 9, base_family = "")
```



```r
df %>% group_by(Value) %>%
  summarize(P05 = quantile(Predicted_logExp, 0.05),
            P50 = median(Predicted_logExp),
            P95 = quantile(Predicted_logExp, 0.95))
```

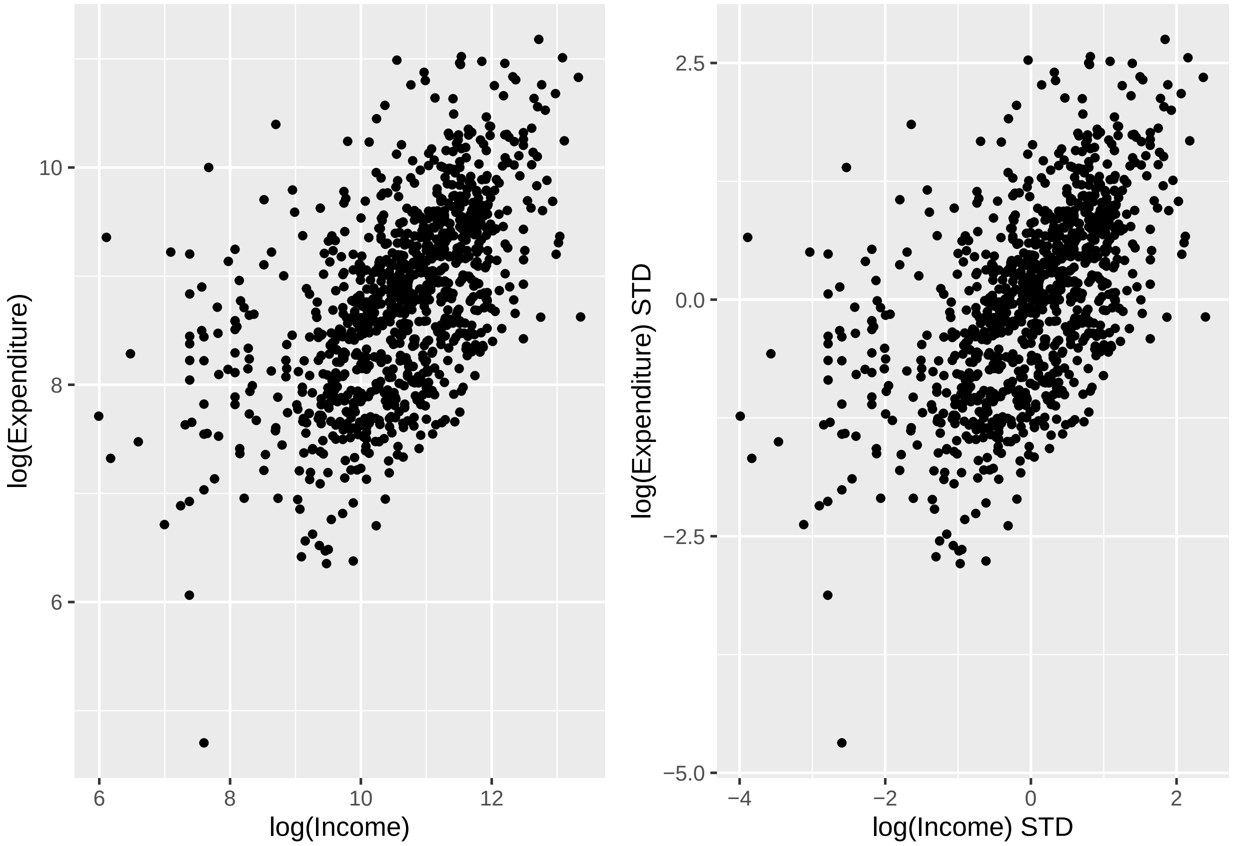
```
## # A tibble: 4 x 4
##   Value               P05   P50   P95
##   <chr>             <dbl> <dbl> <dbl>
## 1 log(Income) = 1   3.50  4.73  5.96
## 2 log(Income) = 5   5.25  6.43  7.67
## 3 log(Income) = 7   6.04  7.27  8.45
## 4 log(Income) = 9   6.93  8.13  9.28
```

More on priors

Subjective prior: standardization

```
CEData$log_TotalExpSTD <- scale(CEData$log_TotalExp)
CEData$log_TotalIncomeSTD <- scale(CEData$log_TotalIncome)
```

```
g2 = ggplot(CEData, aes(x = log_TotalIncomeSTD, y = log_TotalExpSTD)) +
  geom_point(size=1) +
  xlab("log(Income) STD") + ylab("log(Expenditure) STD") +
  theme_grey(base_size = 10, base_family = "")
grid.arrange(g1, g2, ncol=2)
```



Subjective prior: JAGS script for the standardized SLR model

```
modelString <-"
model {
## sampling
for (i in 1:N){
y[i] ~ dnorm(beta0 + beta1*x[i], invsigma2)
}
## priors
beta0 ~ dnorm(mu0, g0)
beta1 ~ dnorm(mu1, g1)
invsigma2 ~ dgamma(a, b)
sigma <- sqrt(pow(invsigma2, -1))
}
"
```

- Pass the data and hyperparameter values to JAGS:

```
y <- as.vector(CEData$log_TotalExpSTD)
x <- as.vector(CEData$log_TotalIncomeSTD)
N <- length(y)
the_data <- list("y" = y, "x" = x, "N" = N,
                 "mu0" = 0, "g0" = 1,
                 "mu1" = 0.7, "g1" = 44.4,
                 "a" = 1, "b" = 1)

initsfunction <- function(chain){
  .RNG.seed <- c(1,2)[chain]
  .RNG.name <- c("base::Super-Duper",
                 "base::Wichmann-Hill")[chain]
  return(list(.RNG.seed=.RNG.seed,
              .RNG.name=.RNG.name))
}
```

- Run the JAGS code for this model:

```
posterior_sub <- run.jags(modelString,
                      n.chains = 1,
                      data = the_data,
                      monitor = c("beta0", "beta1", "sigma"),
                      adapt = 1000,
                      burnin = 5000,
                      sample = 5000,
                      thin = 1,
                      inits = initsfunction)
```

```
## Calling the simulation...
## Welcome to JAGS 4.3.0 on Mon Nov 25 12:22:34 2019
## JAGS is free software and comes with ABSOLUTELY NO WARRANTY
## Loading module: basemod: ok
## Loading module: bugs: ok
## . . Reading data file data.txt
## . Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 994
##    Unobserved stochastic nodes: 3
##    Total graph size: 3466
## . Reading parameter file inits1.txt
## . Initializing model
## . Adaptation skipped: model is not in adaptive mode.
## . Updating 5000
## -------------------------------------------------| 5000
## ************************************************** 100%
## . . . . Updating 5000
## -------------------------------------------------| 5000
## ************************************************** 100%
## . . . . Updating 0
## . Deleting model
## .
## Note: the model did not require adaptation
## Simulation complete.  Reading coda files...
## Coda files loaded successfully
## Calculating summary statistics...
## Finished running the simulation
```

Subjective prior: JAGS output for the SLR model

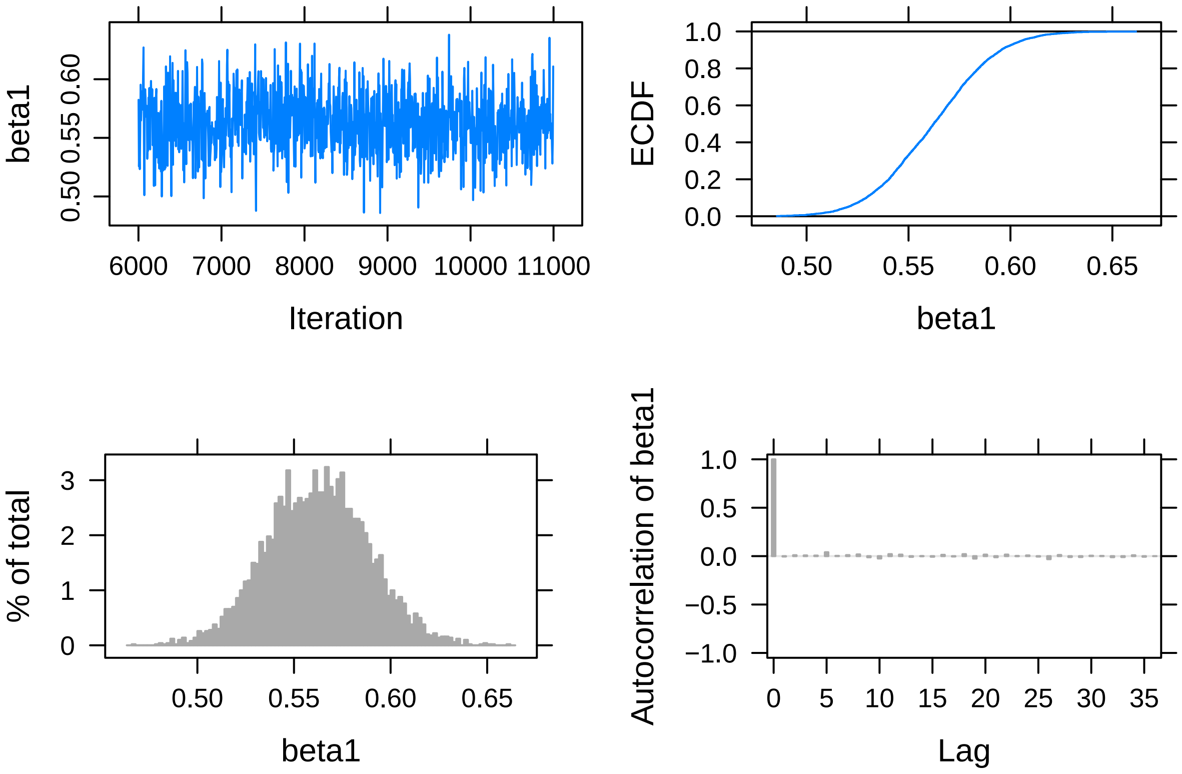
- Obtain posterior summaries of all parameters:

```
summary(posterior_sub)
```

```
##          Lower95        Median   Upper95          Mean         SD Mode
## beta0 -0.0484962 -0.0000908726 0.0544673 -5.334241e-05 0.02637737   NA
## beta1  0.5131890  0.5623200000 0.6144910  5.622705e-01 0.02616385   NA
## sigma  0.7958380  0.8314665000 0.8684720  8.318869e-01 0.01866642   NA
##              MCerr MC%ofSD SSeff       AC.10 psrf
## beta0 0.0003561888     1.4  5484 -0.02064139   NA
## beta1 0.0003700127     1.4  5000 -0.02451218   NA
## sigma 0.0002502895     1.3  5562  0.02083092   NA
```

```
plot(posterior_sub, vars = "beta1")
```

```
## Generating plots...
```



Conditional means prior: JAGS script

```
modelString <-"
model {
## sampling
for (i in 1:N){
y[i] ~ dnorm(beta0 + beta1*x[i], invsigma2)
}

## priors
beta1 <- (mu2 - mu1)/(x2 - x1)
beta0 <- mu1 - x1*(mu2 - mu1)/(x2 - x1)
mu1 ~ dnorm(m1, g1)
mu2 ~ dnorm(m2, g2)
invsigma2 ~ dgamma(a, b)
sigma <- sqrt(pow(invsigma2, -1))
}
"
```

A multiple linear regression, and MCMC simulation by JAGS

JAGS script for the MLR model

```
CEData$log_TotalExpSTD <- scale(CEData$log_TotalExp)
CEData$log_TotalIncomeSTD <- scale(CEData$log_TotalIncome)

library(fastDummies)
## create indictor variable for Rural
CEData$Rural = fastDummies::dummy_cols(CEData$UrbanRural)[,names(fastDummies::dummy_cols(CEData
 == ".data_2"]
```

```
## create indicator variables for Black (2), Native American (3),
## Asian (4), Pacific Islander (5), and Multi-race (6)
CEData$Race_Black = fastDummies::dummy_cols(CEData$Race)[,names(fastDummies::dummy_cols(CEData$
CEData$Race_NA = fastDummies::dummy_cols(CEData$Race)[,names(fastDummies::dummy_cols(CEData$Rac
CEData$Race_Asian = fastDummies::dummy_cols(CEData$Race)[,names(fastDummies::dummy_cols(CEData$
CEData$Race_PI = fastDummies::dummy_cols(CEData$Race)[,names(fastDummies::dummy_cols(CEData$Rac
CEData$Race_M = fastDummies::dummy_cols(CEData$Race)[,names(fastDummies::dummy_cols(CEData$Race
```

```
modelString <-"
model {
## sampling
for (i in 1:N){
y[i] ~ dnorm(beta0 + beta1*x_income[i] + beta2*x_rural[i] +
beta3*x_race_B[i] + beta4*x_race_N[i] +
beta5*x_race_A[i] + beta6*x_race_P[i] +
beta7*x_race_M[i], invsigma2)
}
## priors
beta0 ~ dnorm(mu0, g0)
beta1 ~ dnorm(mu1, g1)
beta2 ~ dnorm(mu2, g2)
beta3 ~ dnorm(mu3, g3)
beta4 ~ dnorm(mu4, g4)
beta5 ~ dnorm(mu5, g5)
beta6 ~ dnorm(mu6, g6)
beta7 ~ dnorm(mu7, g7)
invsigma2 ~ dgamma(a, b)
sigma <- sqrt(pow(invsigma2, -1))
}
"
```

- Pass the data and hyperparameter values to JAGS:

```
y = as.vector(CEData$log_TotalExpSTD)
x_income = as.vector(CEData$log_TotalIncomeSTD)
x_rural = as.vector(CEData$Rural)
```

```
x_race_B = as.vector(CEData$Race_Black)
x_race_N = as.vector(CEData$Race_NA)
x_race_A = as.vector(CEData$Race_Asian)
x_race_P = as.vector(CEData$Race_PI)
x_race_M = as.vector(CEData$Race_M)
N = length(y)  # Compute the number of observations
```

- Pass the data and hyperparameter values to JAGS:

```
the_data <- list("y" = y, "x_income" = x_income,
                 "x_rural" = x_rural, "x_race_B" = x_race_B,
                 "x_race_N" = x_race_N, "x_race_A" = x_race_A,
                 "x_race_P" = x_race_P, "x_race_M" = x_race_M,
                 "N" = N,
                 "mu0" = 0, "g0" = 1, "mu1" = 0, "g1" = 1,
                 "mu2" = 0, "g2" = 1, "mu3" = 0, "g3" = 1,
                 "mu4" = 0, "g4" = 1, "mu5" = 0, "g5" = 1,
                 "mu6" = 0, "g6" = 1, "mu7" = 0, "g7" = 1,
                 "a" = 1, "b" = 1)
```

- Pass the data and hyperparameter values to JAGS:

```
initsfunction <- function(chain){
  .RNG.seed <- c(1,2)[chain]
  .RNG.name <- c("base::Super-Duper",
                 "base::Wichmann-Hill")[chain]
  return(list(.RNG.seed=.RNG.seed,
              .RNG.name=.RNG.name))
}
```

- Run the JAGS code for this model:

```
posterior_MLR <- run.jags(modelString,
                          n.chains = 1,
                          data = the_data,
                          monitor = c("beta0", "beta1", "beta2",
                                      "beta3", "beta4", "beta5",
                                      "beta6", "beta7", "sigma"),
                          adapt = 1000,
                          burnin = 5000,
                          sample = 5000,
                          thin = 1,
                          inits = initsfunction)
```

```
## Calling the simulation...
## Welcome to JAGS 4.3.0 on Mon Nov 25 12:22:38 2019
## JAGS is free software and comes with ABSOLUTELY NO WARRANTY
## Loading module: basemod: ok
## Loading module: bugs: ok
## . . Reading data file data.txt
## . Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 994
##    Unobserved stochastic nodes: 9
##    Total graph size: 9529
## . Reading parameter file inits1.txt
## . Initializing model
## . Adaptation skipped: model is not in adaptive mode.
## . Updating 5000
## -------------------------------------------------| 5000
## ************************************************** 100%
## . . . . . . . . . . Updating 5000
## -------------------------------------------------| 5000
## ************************************************** 100%
## . . . . Updating 0
## . Deleting model
## .
## Note: the model did not require adaptation
## Simulation complete.  Reading coda files...
## Coda files loaded successfully
## Calculating summary statistics...

## Warning: Convergence cannot be assessed with only 1 chain

## Finished running the simulation
```

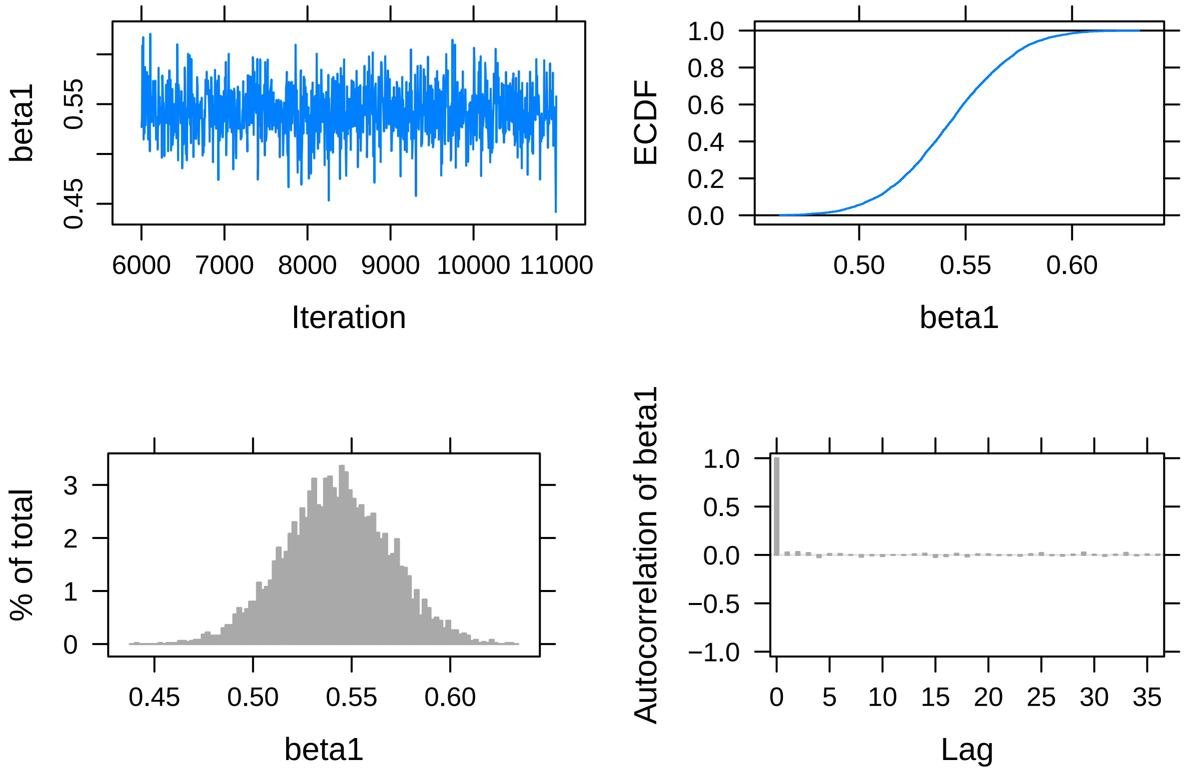
JAGS output for the MLR model

```
summary(posterior_MLR)
```

```
##          Lower95      Median    Upper95        Mean          SD Mode
## beta0 -0.0261884  0.03072615  0.0898729  0.03111402 0.02974211   NA
## beta1  0.4889750  0.54241750  0.5926740  0.54237419 0.02648363   NA
## beta2 -0.4918380 -0.26816300 -0.0262582 -0.26872848 0.11800343   NA
## beta3 -0.3996750 -0.23509350 -0.0678321 -0.23338067 0.08548032   NA
## beta4 -0.6071250 -0.01422395  0.5520270 -0.01764766 0.29634497   NA
## beta5 -0.0922392  0.18444550  0.4434330  0.18556012 0.13734992   NA
## beta6 -0.5323690  0.08998935  0.6842530  0.09247921 0.31739975   NA
## beta7 -0.3609250  0.03685400  0.4281700  0.03163386 0.20378584   NA
## sigma  0.7892530  0.82702250  0.8624090  0.82749520 0.01860583   NA
##              MCerr MC%ofSD SSeff        AC.10 psrf
## beta0 0.0005408114     1.8  3024 -0.004175826   NA
## beta1 0.0003934179     1.5  4532 -0.012275215   NA
## beta2 0.0018827201     1.6  3928 -0.001199134   NA
## beta3 0.0014087613     1.6  3682 -0.006798767   NA
## beta4 0.0041909508     1.4  5000 -0.024524053   NA
## beta5 0.0020743101     1.5  4384 -0.007574647   NA
## beta6 0.0044887103     1.4  5000  0.007878188   NA
## beta7 0.0030045388     1.5  4600  0.011216547   NA
## sigma 0.0002631261     1.4  5000  0.011646253   NA
```

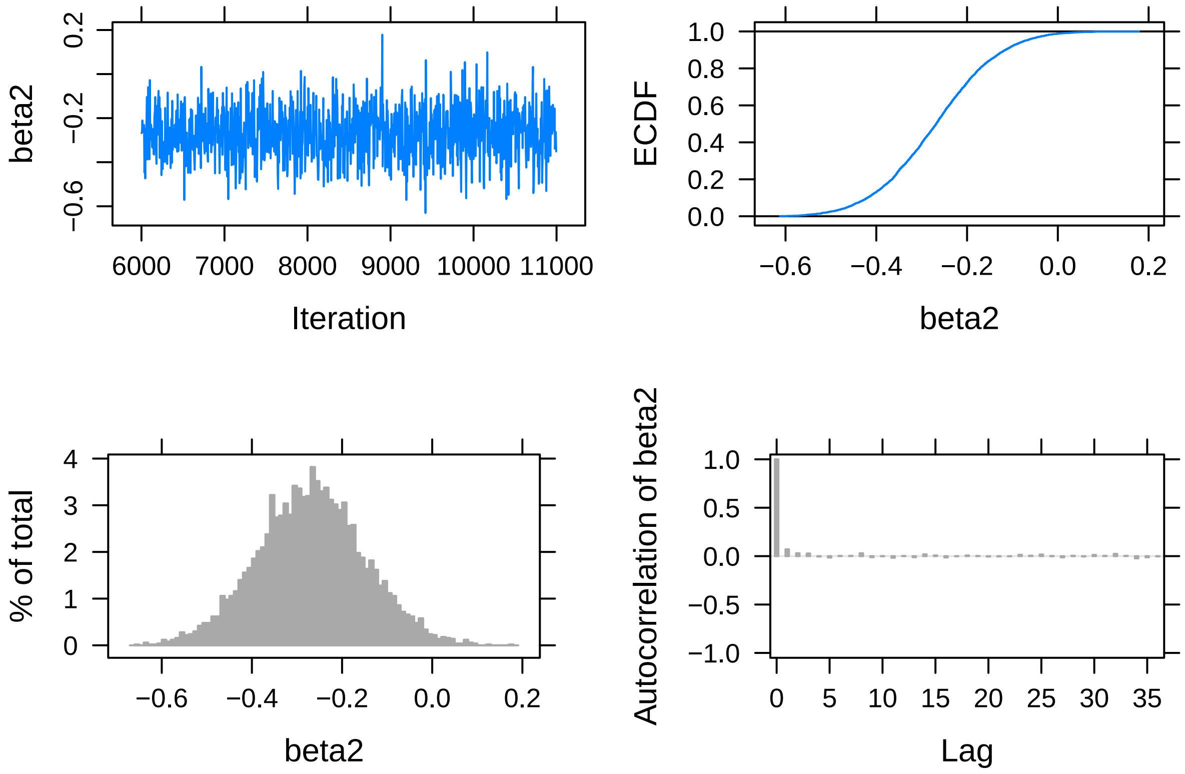
```
plot(posterior_MLR, vars = "beta1")
```

```
## Generating plots...
```



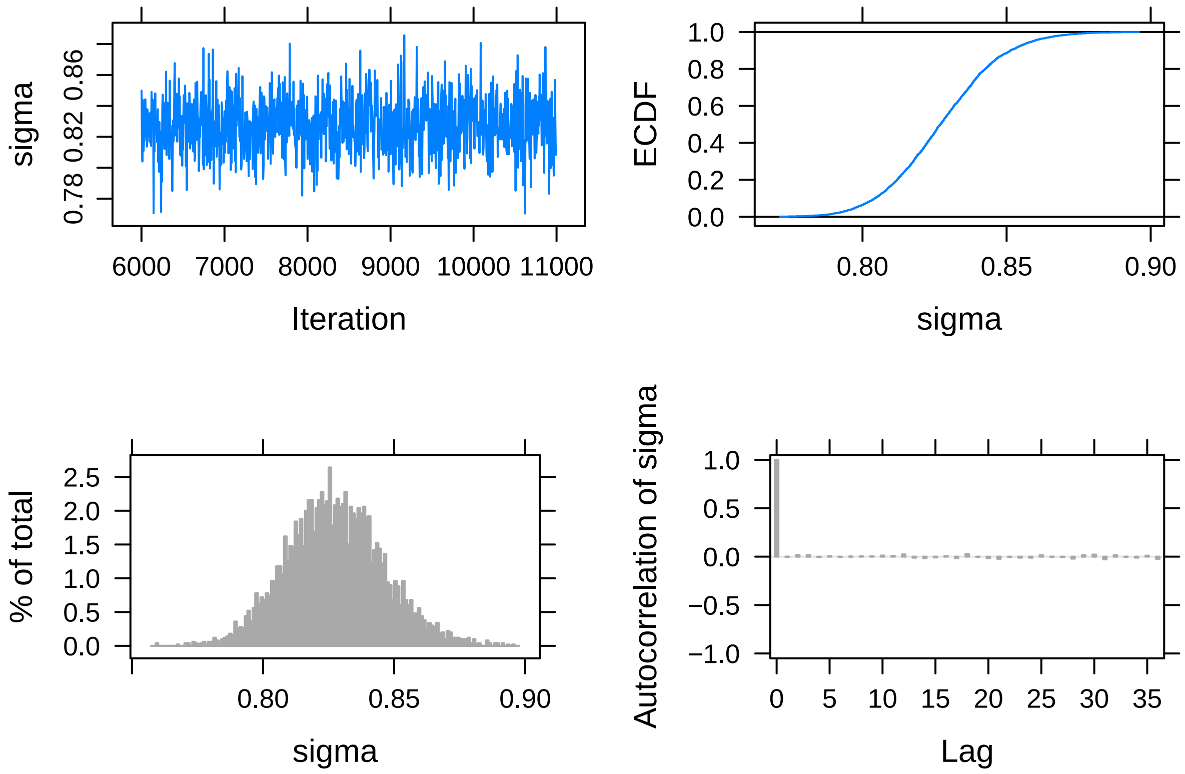
```
plot(posterior_MLR, vars = "beta2")
```

```
## Generating plots...
```

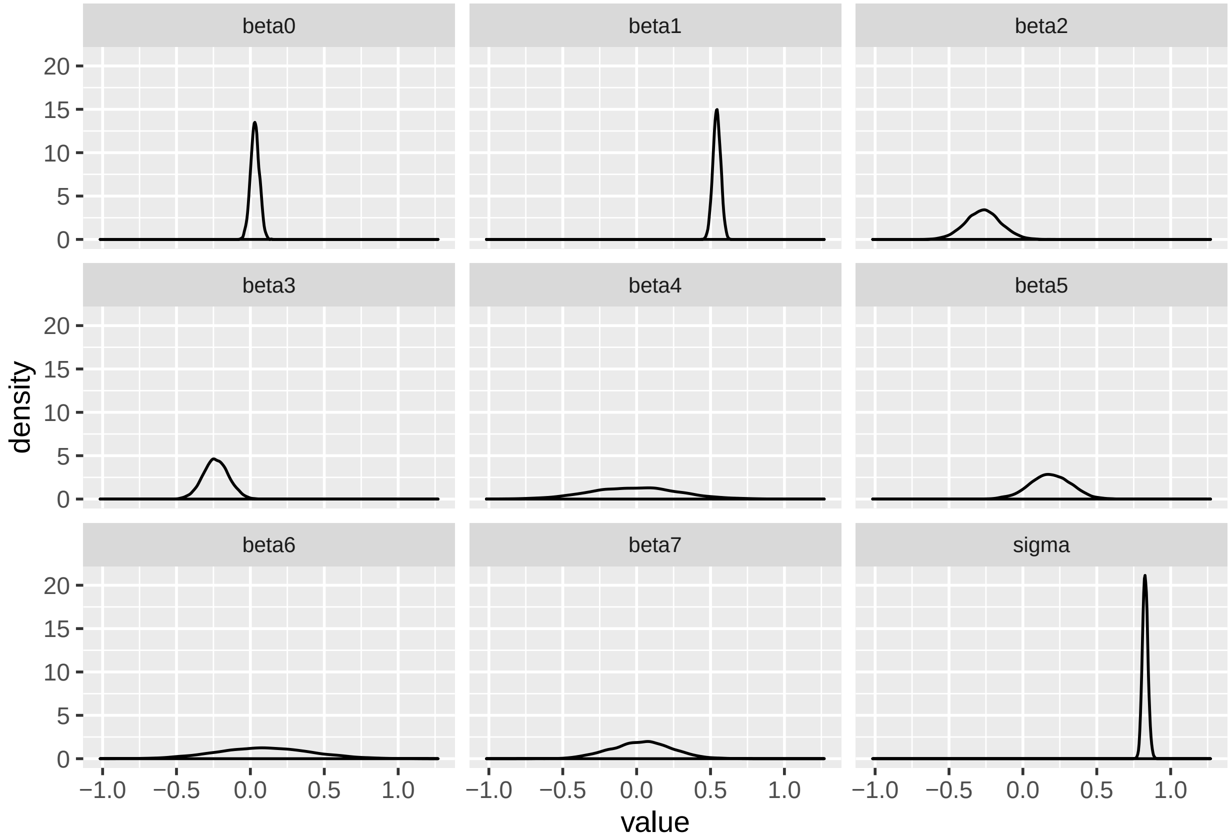


```r
plot(posterior_MLR, vars = "sigma")
```

```
## Generating plots...
```

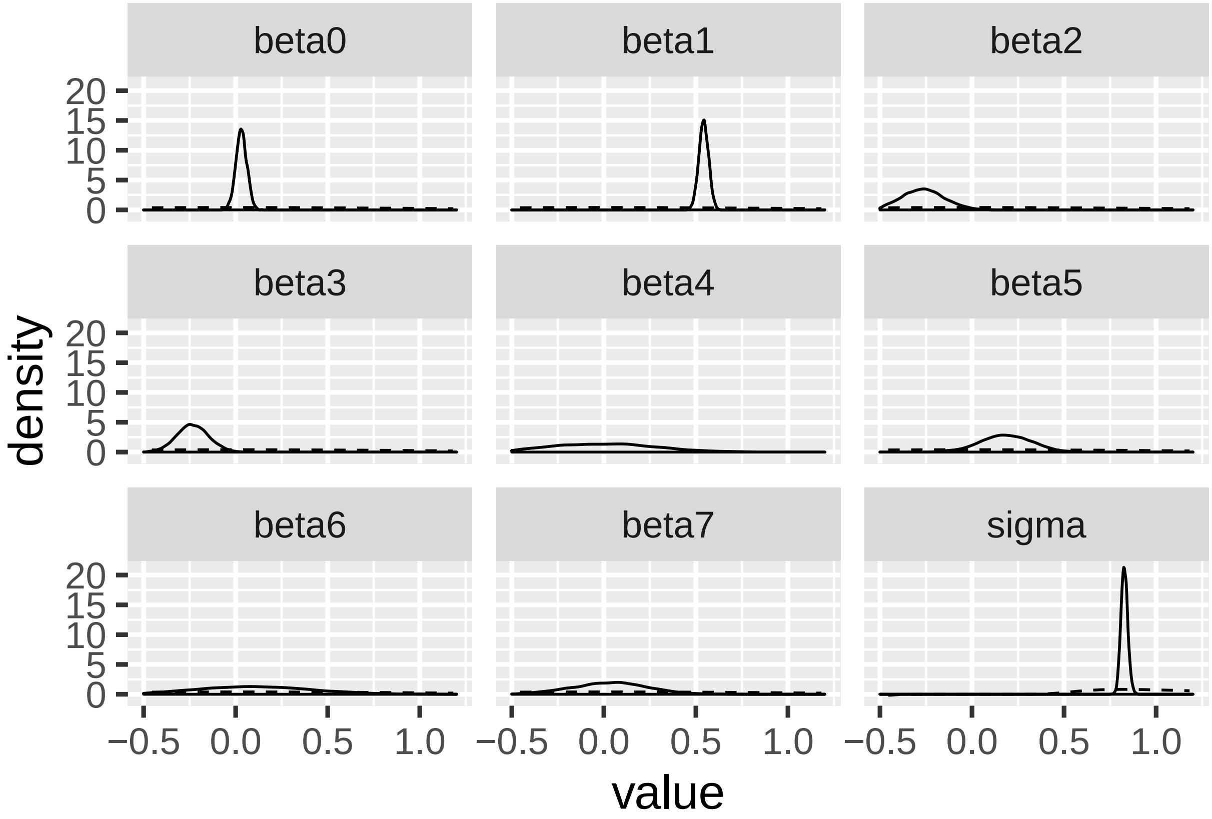


```
post <- as.mcmc(posterior_MLR)
post %>% as.data.frame %>%
  gather(parameter, value) -> post2
ggplot(post2, aes(value)) +
  geom_density() + facet_wrap(~ parameter, ncol = 3) +
  theme(strip.text.x = element_text(size=8))
```



```
## Warning: Removed 548 rows containing non-finite values (stat_density).

## Warning: Removed 72 rows containing missing values (geom_path).
```


beta0
beta1
beta2
beta3
beta4
beta5
beta6
beta7
sigma
density
20
15
10
5
0
−0.5
0.0
0.5
1.0
value