IPUMS Health Data

MATH 301 Data Confidentiality

Henrik Olsson

February 25, 2020

```r
ipumsdata1<- read.csv("nhis_00001.csv")
ipumsdata1$logincome <- log(ipumsdata1$EARNIMPOINT1)

ipumsdata <- sample_n(ipumsdata1, 10000, replace = FALSE, prob = NULL)
ipumsdata <- ipumsdata[!ipumsdata$EARNIMPOINT1 ==0, ]
ipumsdata$logincome<- log(ipumsdata$EARNIMPOINT1)
head(ipumsdata)
```

```
##    AGE SEX RACEA EDUCREC2 HOURSWRK POORYN EARNIMP1 EARNIMPOINT1 USUALPL
## 3   40   1   100       42       37      1       31        48000       2
## 9   29   2   100       54       40      1       12        30000       1
## 11  65   1   100       51       40      1       51        65000       2
## 12  27   2   100       51       32      1       12        30000       0
## 15  37   1   100       51       32      1        4        16726       2
## 18  33   1   100       60       40      1       31        48000       0
##    DELAYCOST HINOTCOVE ALCDAYSWK CIGDAYMO HRSLEEP WORFREQ DEPFREQ
## 3          1         1        30       96       8       4       5
## 9          1         2         0       96       7       5       5
## 11         1         1        96       96       8       5       5
## 12         1         1        96       96       0       0       0
## 15         1         2         0       96       7       5       5
## 18         1         1        96       96       0       0       0
##    logincome
## 3   10.77896
## 9   10.30895
## 11  11.08214
## 12  10.30895
## 15   9.72472
## 18  10.77896
```

Our goal is to generate synthetic data from the estimated Bayesian synthesizer from the posterior predictive distribution. To produce a good synthesizer, there will be trade-offs between utility and risks.

The two most sensitive variables are a person's imputed total earnings from the previous calender year and total hours worked last week or usually. The latter contains 99 categories, while the former contains 70 categories. If an intruder were to know one's total earnings or amount of work time then they can obtain the person's information with much greater probability than if they had access to another variable.

Measuring log income with respect to frequency of alcohol drank, how often one feels anxious, and health care coverage

First, lets look at the relationship between frequency drank alcohol in past year, how often feel worried, nervous, or anxious, and health care coverage.

```r
## JAGS script
modelString <-"
model {
```

```
## sampling
for (i in 1:N){
y[i] ~ dnorm(beta0 + beta1*x_alc_one[i] +
beta2*x_alc_two[i] + beta3*x_alc_three[i] +
beta4*x_alc_four[i] + beta5*x_alc_five[i] +
beta6*x_alc_six[i] + beta7*x_alc_seven[i] +
beta8*x_alc_none[i] + beta9*x_wor_daily[i] +
beta10*x_wor_weekly[i] + beta11*x_wor_monthly[i] +
beta12*x_wor_fewtimes[i] + beta13*x_wor_never[i] +
beta14*x_health_cov[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)
beta8 ~ dnorm(mu8, g8)
beta9 ~ dnorm(mu9, g9)
beta10 ~ dnorm(mu10, g10)
beta11 ~ dnorm(mu11, g11)
beta12 ~ dnorm(mu12, g12)
beta13 ~ dnorm(mu13, g13)
beta14 ~ dnorm(mu14, g14)
invsigma2 ~ dgamma(a, b)
sigma <- sqrt(pow(invsigma2, -1))
}"
```

```
y = as.vector(ipumsdata$logincome)
x_alc_one = as.vector(ipumsdata$ALC$.data_0)
x_alc_two = as.vector(ipumsdata$ALC$.data_80)
x_alc_three = as.vector(ipumsdata$ALC$.data_96)
x_alc_four = as.vector(ipumsdata$ALC$.data_10)
x_alc_five = as.vector(ipumsdata$ALC$.data_70)
x_alc_six = as.vector(ipumsdata$ALC$.data_30)
x_alc_seven = as.vector(ipumsdata$ALC$.data_50)
x_alc_none = as.vector(ipumsdata$ALC$.data_20)
x_wor_daily = as.vector(ipumsdata$WORRY$.data_0)
x_wor_weekly = as.vector(ipumsdata$WORRY$.data_5)
x_wor_monthly = as.vector(ipumsdata$WORRY$.data_1)
x_wor_fewtimes = as.vector(ipumsdata$WORRY$.data_2)
x_wor_never = as.vector(ipumsdata$WORRY$.data_4)
x_health_cov = as.vector(ipumsdata$HEALTH$.data_1)
N = length(y) # Compute the number of observations


## Pass the data and hyperparameter values to JAGS
the_data <- list("y" = y,
"x_alc_one" = x_alc_one, "x_alc_two" = x_alc_two,
"x_alc_three" = x_alc_three, "x_alc_four" = x_alc_four,
"x_alc_five" = x_alc_five, "x_alc_six" = x_alc_six,
```

```r
"x_alc_seven" = x_alc_seven, "x_alc_none" = x_alc_none,
"x_wor_daily" = x_wor_daily, "x_wor_weekly" = x_wor_weekly,
"x_wor_monthly" = x_wor_monthly, "x_wor_fewtimes" = x_wor_fewtimes,
"x_wor_never" = x_wor_never, "x_health_cov" = x_health_cov, "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,
"mu8" = 0, "g8" = 1, "mu9" = 0, "g9" = 1,
"mu10" = 0, "g10" = 1, "mu11" = 0, "g11" = 1,
"mu12" = 0, "g12" = 1, "mu13" = 0, "g13" = 1,
"mu14" = 0, "g14" = 1, "a" = 1, "b" = 1)
```

```r
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))
}
```

```r
## 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", "beta8", "beta9", "beta10",
"beta11", "beta12", "beta13", "beta14", "sigma"),
adapt = 1000,
burnin = 5000,
sample = 5000,
thin = 1,
inits = initsfunction)
```

```
## Loading required namespace: rjags

## Compiling rjags model...
## Calling the simulation using the rjags method...
## Note: the model did not require adaptation
## Burning in the model for 5000 iterations...
## Running the model for 5000 iterations...
## Simulation complete
## Calculating summary statistics...
## Finished running the simulation
```

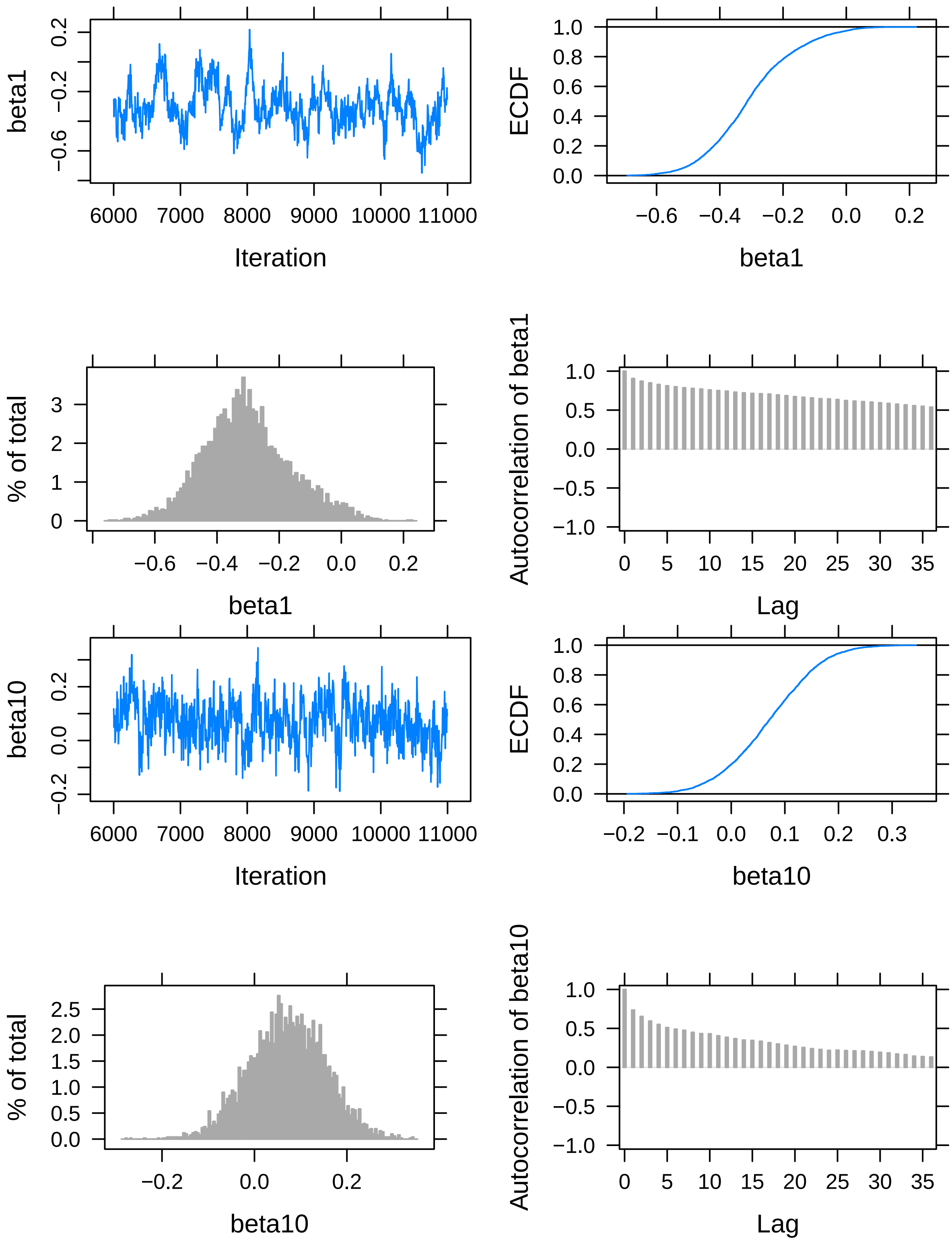
```r
## JAGS output
summary(posterior_MLR)
```

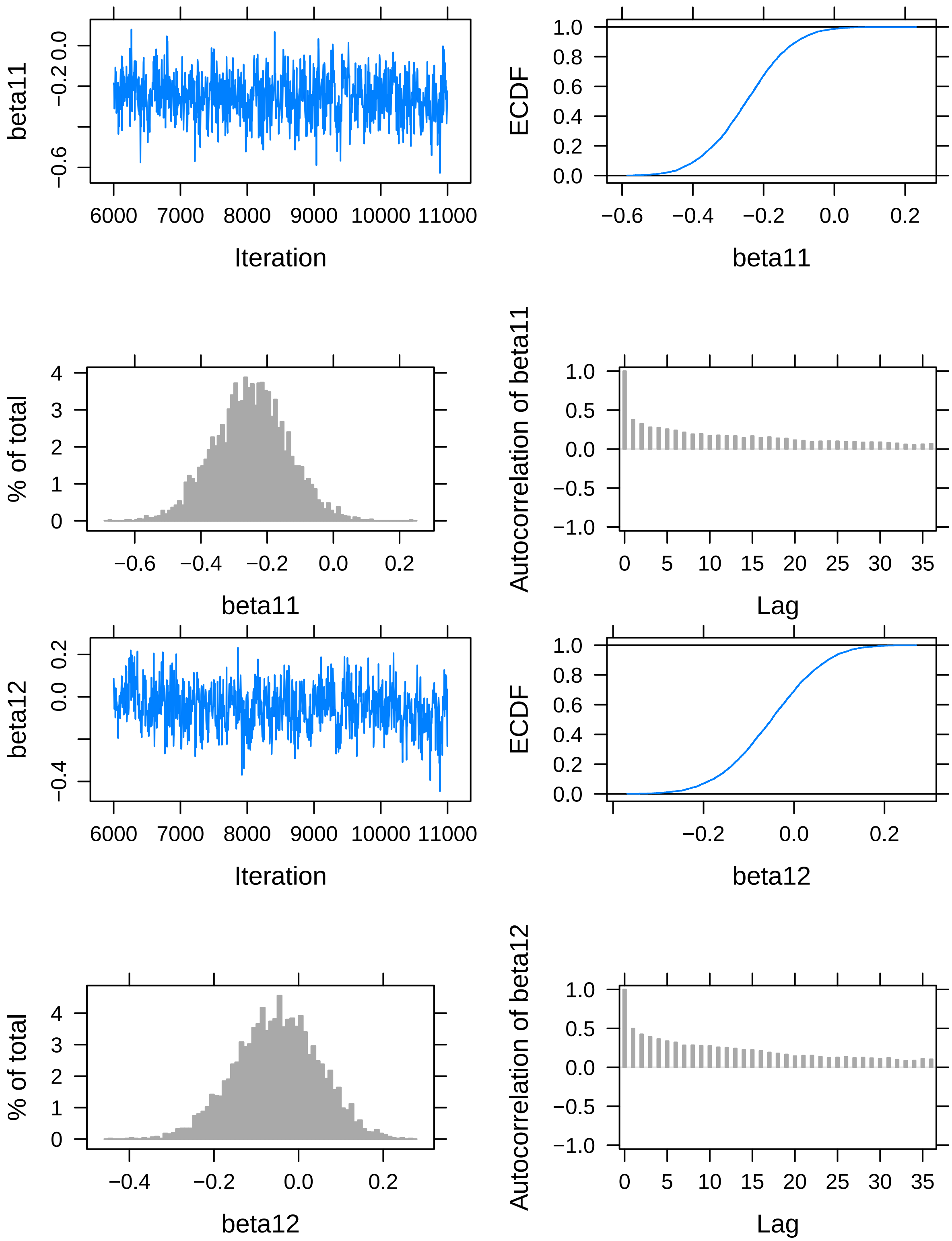
```
##            Lower95      Median      Upper95        Mean         SD Mode
## beta0   9.78893861 10.08397270 10.388911445 10.07750038 0.15444469   NA
## beta1  -0.55489106 -0.31311524  0.002326204 -0.30334584 0.13961593   NA
## beta2  -0.86499298 -0.57794665 -0.257269351 -0.56951692 0.15362023   NA
## beta3  -1.08066368 -0.76956123 -0.457680351 -0.75731213 0.15746751   NA
## beta4  -0.33205392 -0.04927050  0.251884935 -0.03929082 0.14808760   NA
## beta5  -0.39740343 -0.07184949  0.284090243 -0.06503680 0.17300271   NA
```

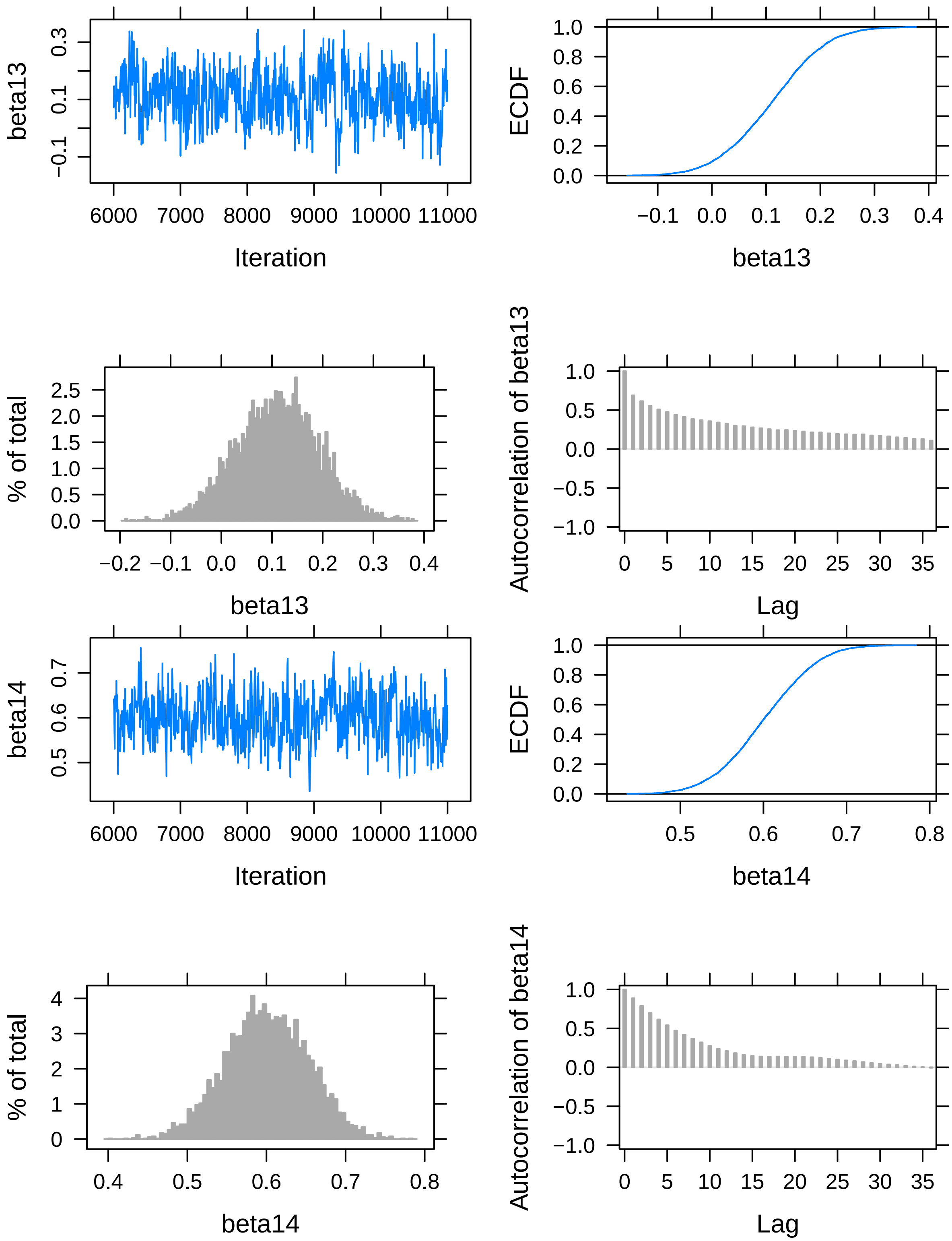
```
## beta6  -0.25590330  0.08237363  0.408156744  0.08868493 0.16866515   NA
## beta7  -0.15642414  0.26000893  0.712932394  0.26524430 0.22342618   NA
## beta8  -0.24150582  0.02576994  0.366657275  0.03509290 0.15687133   NA
## beta9   0.22986653  0.43155559  0.627717508  0.43286416 0.09995429   NA
## beta10 -0.08179111  0.07143744  0.237851382  0.07078029 0.08273579   NA
## beta11 -0.46407857 -0.24849825 -0.034922020 -0.24946581 0.11086535   NA
## beta12 -0.24858503 -0.04949193  0.128712515 -0.05159794 0.09844792   NA
## beta13 -0.05017867  0.11195497  0.274435896  0.11055060 0.08375184   NA
## beta14  0.50180083  0.59992816  0.702603821  0.60057530 0.05239905   NA
## sigma   1.17615807  1.20091732  1.223360376  1.20096479 0.01198080   NA
##               MCerr MC%ofSD SSeff      AC.10 psrf
## beta0  0.0282792101    18.3    30 0.88346057   NA
## beta1  0.0228682530    16.4    37 0.75978035   NA
## beta2  0.0232995067    15.2    43 0.63896121   NA
## beta3  0.0246446226    15.7    41 0.85289666   NA
## beta4  0.0220383322    14.9    45 0.67166414   NA
## beta5  0.0219089106    12.7    62 0.51210798   NA
## beta6  0.0205262885    12.2    68 0.52835053   NA
## beta7  0.0217917898     9.8   105 0.28890250   NA
## beta8  0.0235953939    15.0    44 0.59124951   NA
## beta9  0.0088981895     8.9   126 0.61619672   NA
## beta10 0.0062429722     7.5   176 0.43054468   NA
## beta11 0.0055017649     5.0   406 0.17150939   NA
## beta12 0.0058528584     5.9   283 0.27545990   NA
## beta13 0.0054723854     6.5   234 0.35786780   NA
## beta14 0.0030334075     5.8   298 0.27622685   NA
## sigma  0.0001731841     1.4  4786 0.01211874   NA
```

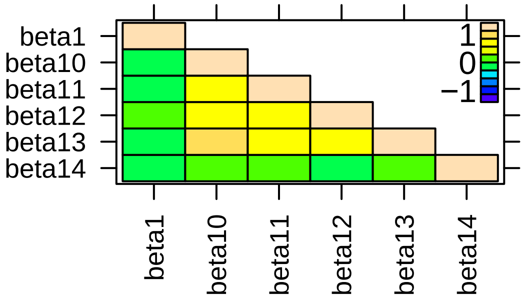
```r
plot(posterior_MLR, vars = "beta1")
```

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


beta1
Iteration
ECDF
beta1
% of total
beta1
Autocorrelation of beta1
Lag
beta10
Iteration
ECDF
beta10
% of total
beta10
Autocorrelation of beta10
Lag

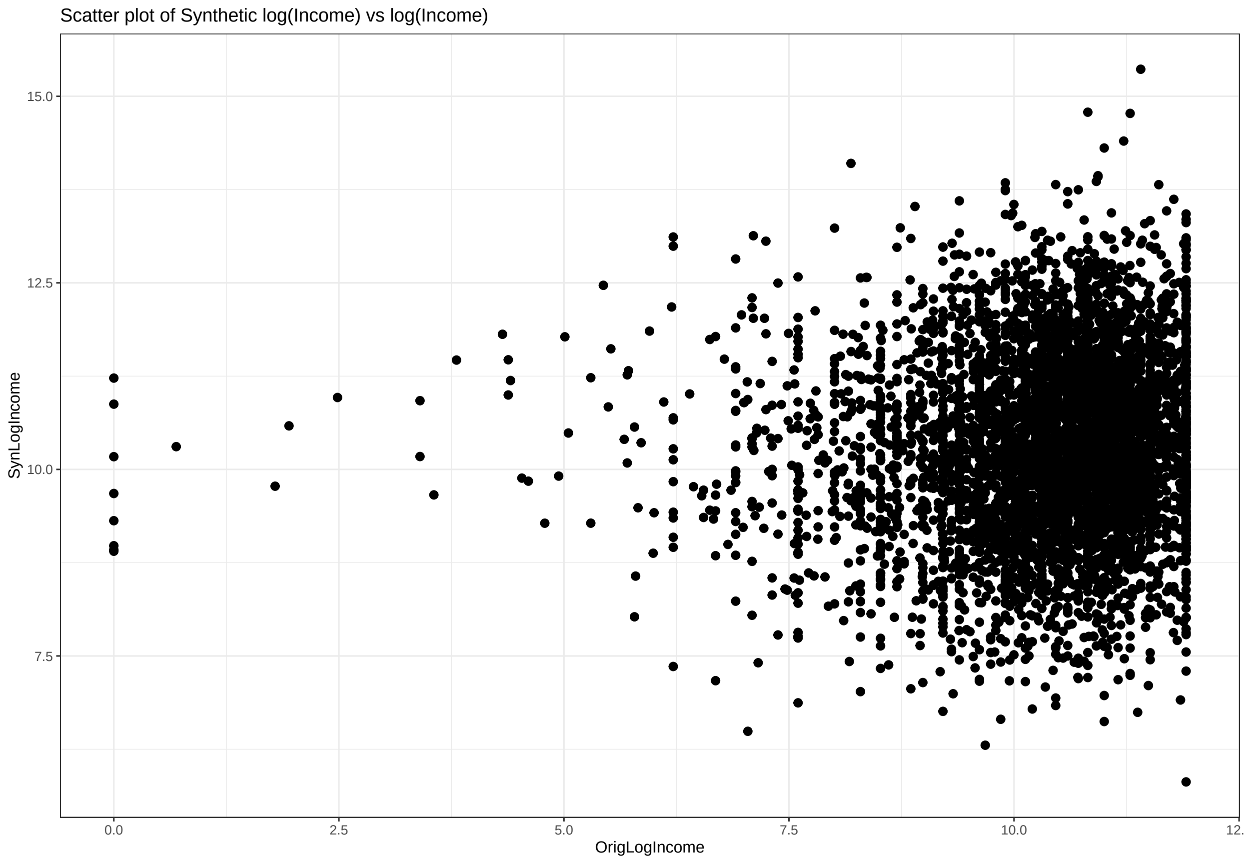

beta11
Iteration
6000 7000 8000 9000 10000 11000
ECDF
beta11
% of total
Autocorrelation of beta11
Lag
beta12
Iteration
ECDF
beta12
% of total
beta12
Autocorrelation of beta12
Lag


beta13
Iteration
ECDF
beta13
% of total
beta13
Autocorrelation of beta13
Lag
beta14
Iteration
ECDF
beta14
% of total
beta14
Autocorrelation of beta14
Lag



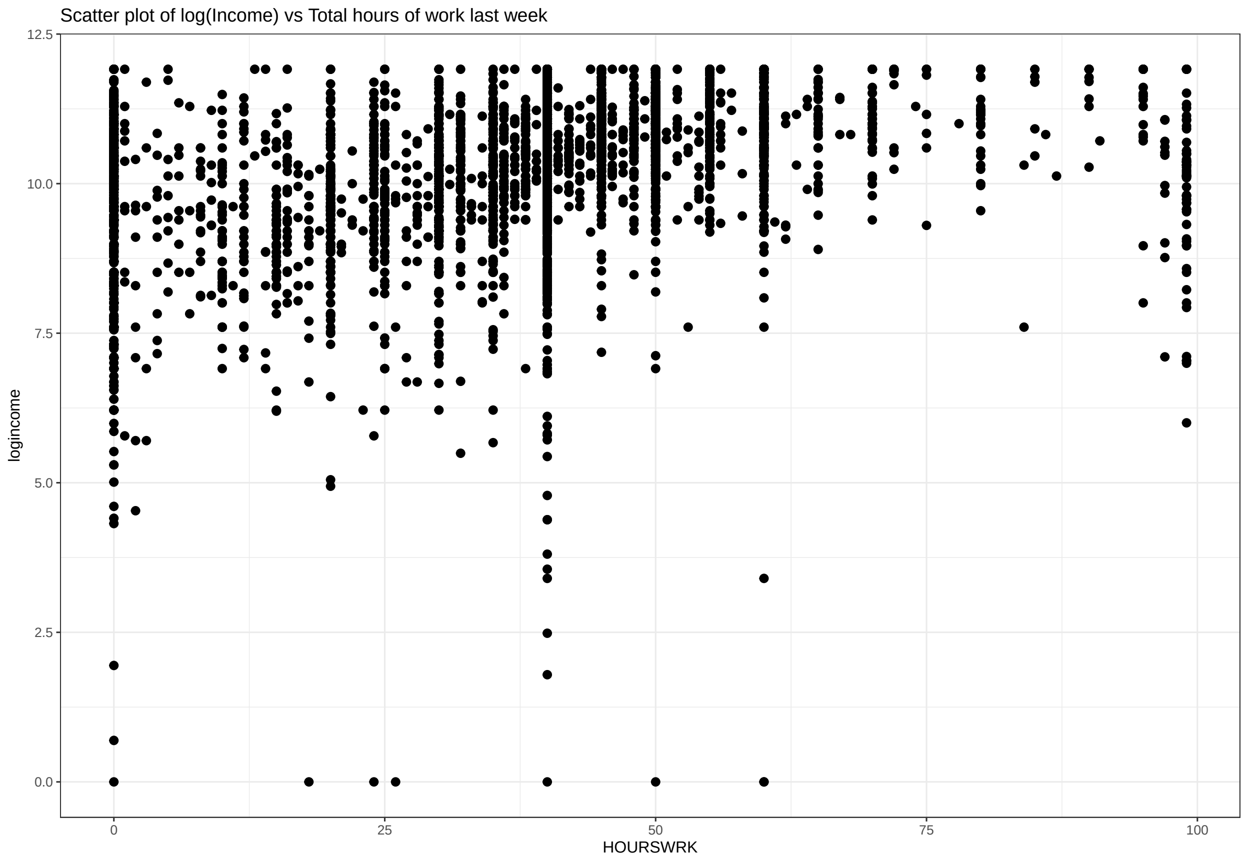

```
## Saving posterior parameter draws
post <- as.mcmc(posterior_MLR)

## Generating one set of sythetic data
synthesize <- function(X, index, n){
  mean_Y <- post[index, "beta0"] +  X$x_alc_one * post[index, "beta1"] +  X$x_alc_two * post[index, "be
  synthetic_Y <- rnorm(n, mean_Y, post[index, "sigma"])
  data.frame(X$y, synthetic_Y)
}
n <- dim(ipumsdata)[1]
new <- data.frame(y, x_alc_one, x_alc_two, x_alc_three, x_alc_four, x_alc_five, x_alc_six, x_alc_seven,
synthetic_one <- synthesize(new, 1, n)
names(synthetic_one) <- c("OrigLogIncome", "SynLogIncome")
```

```
ggplot(synthetic_one, aes(x = OrigLogIncome, y = SynLogIncome)) +
  geom_point(size = 1) +
  labs(title = "Scatter plot of Synthetic log(Income) vs log(Income)") +
  theme_bw(base_size = 6, base_family = "")
```

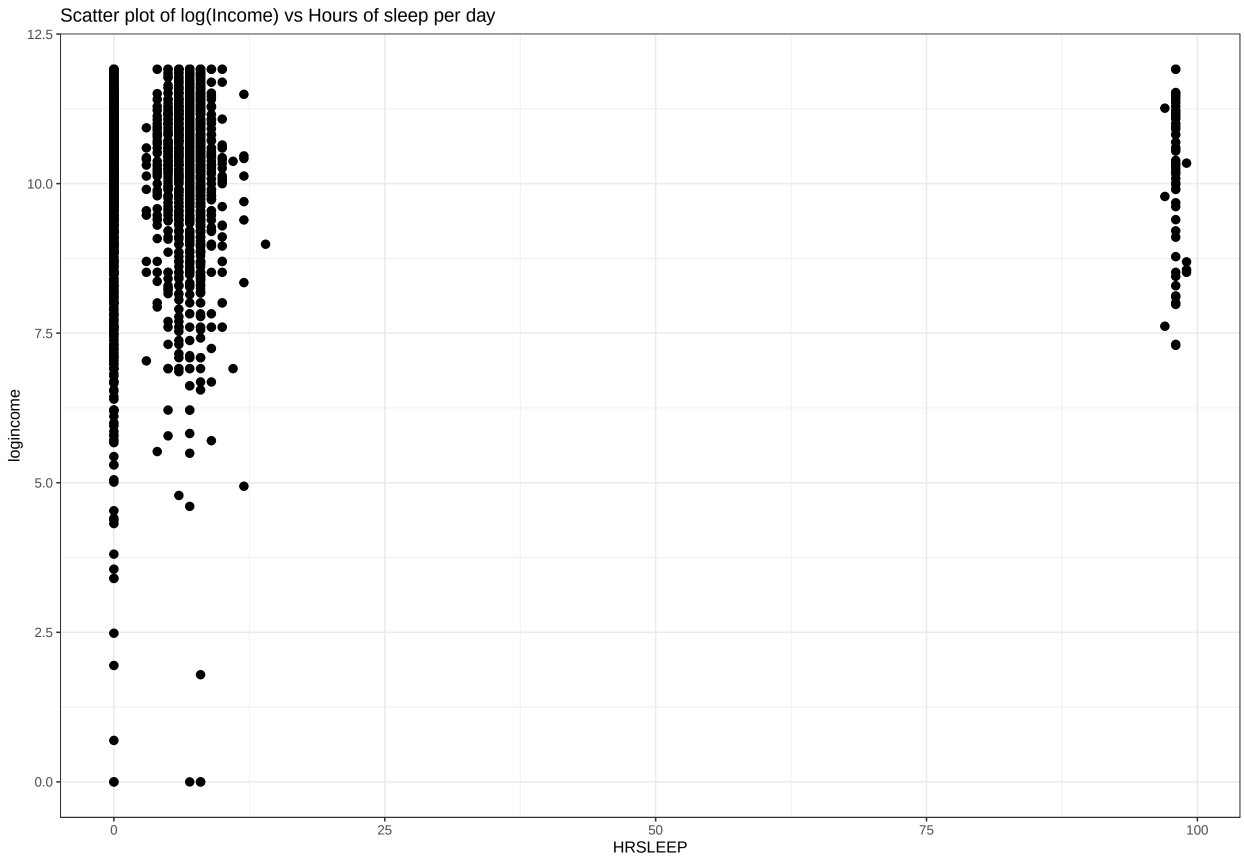


Measure log income with respect to total hours worked last week, hours of sleep, and age

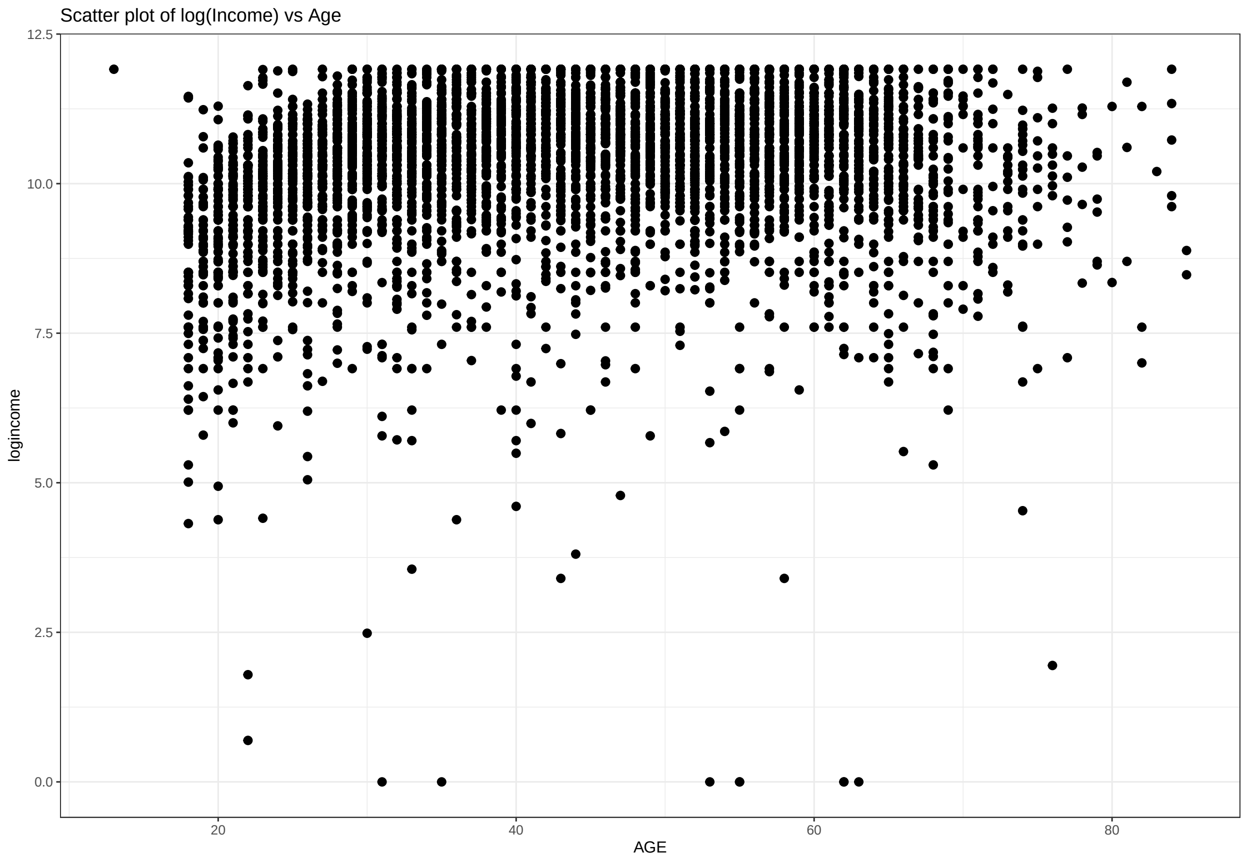
```
ggplot(ipumsdata, aes(x=HOURSWRK, y=logincome)) +
  geom_point(size = 1) +
  labs(title = "Scatter plot of log(Income) vs Total hours of work last week") +
  theme_bw(base_size = 6, base_family = "")
```




```
ggplot(ipumsdata, aes(x=HRSLEEP, y=logincome)) +
  geom_point(size = 1) +
  labs(title = "Scatter plot of log(Income) vs Hours of sleep per day") +
  theme_bw(base_size = 6, base_family = "")
```




```r
ggplot(ipumsdata, aes(x=AGE, y=logincome)) +
  geom_point(size = 1) +
  labs(title = "Scatter plot of log(Income) vs Age") +
  theme_bw(base_size = 6, base_family = "")
```




```
## JAGS script
modelString <-"
model {
## sampling
for (i in 1:N){
y[i] ~ dnorm(beta0 + beta1*x_hrs_work[i] + beta2*x_hrs_sleep[i] + beta3*x_age[i], invsigma2)
}
## priors
beta0 ~ dnorm(mu0, g0)
beta1 ~ dnorm(mu1, g1)
beta2 ~ dnorm(mu2, g2)
beta3 ~ dnorm(mu3, g3)
invsigma2 ~ dgamma(a, b)
sigma <- sqrt(pow(invsigma2, -1))
}
"
```

```
y = as.vector(ipumsdata$logincome)
x_hrs_work = as.vector(ipumsdata$HOURSWRK)
x_hrs_sleep = as.vector(ipumsdata$HRSLEEP)
x_age = as.vector(ipumsdata$AGE)
N = length(y) # Compute the number of observations


## Pass the data and hyperparameter values to JAGS
the_data <- list("y" = y,
"x_hrs_work" = x_hrs_work, "x_hrs_sleep" = x_hrs_sleep,
```

```
"x_age" = x_age, "N" = N,
"mu0" = 0, "g0" = 1, "mu1" = 0, "g1" = 1,
"mu2" = 0, "g2" = 1, "mu3" = 0, "g3" = 1,
 "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_hrswork <- run.jags(modelString,
n.chains = 1,
data = the_data,
monitor = c("beta0", "beta1", "beta2", "beta3", "sigma"),
adapt = 1000,
burnin = 5000,
sample = 5000,
thin = 1,
inits = initsfunction)
```

```
## Compiling rjags model...
## Calling the simulation using the rjags method...
## Note: the model did not require adaptation
## Burning in the model for 5000 iterations...
## Running the model for 5000 iterations...
## Simulation complete
## Calculating summary statistics...

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

## Finished running the simulation
```

```
## JAGS output
summary(posterior_hrswork)
```

```
##            Lower95       Median     Upper95         Mean           SD Mode
## beta0  8.554889038 8.6583652465 8.776021025 8.6607547021 0.0569867593   NA
## beta1  0.023576898 0.0253260291 0.026882643 0.0253095462 0.0008484914   NA
## beta2 -0.002396765 0.0003402787 0.003111094 0.0003462635 0.0014073664   NA
## beta3  0.014227569 0.0163954123 0.018268011 0.0163891221 0.0010276869   NA
## sigma  1.100863943 1.1218618708 1.144395970 1.1219418603 0.0113099746   NA
##              MCerr MC%ofSD SSeff       AC.10 psrf
## beta0 3.848287e-03     6.8   219  0.39302998   NA
## beta1 3.464115e-05     4.1   600  0.09241881   NA
## beta2 2.167880e-05     1.5  4214 -0.01706925   NA
## beta3 6.029627e-05     5.9   290  0.32950421   NA
## sigma 1.599472e-04     1.4  5000  0.01837566   NA
```

```
## Saving posterior parameter draws
post <- as.mcmc(posterior_MLR)

## Generating one set of sythetic data
synthesize <- function(X, index, n){
```

```
  mean_Y <- post[index, "beta0"] +  X$x_hrs_work * post[index, "beta1"] +  X$x_hrs_sleep * post[index,
  synthetic_Y <- rnorm(n, mean_Y, post[index, "sigma"])
  data.frame(X$y, synthetic_Y)
}
n <- dim(ipumsdata)[1]
new <- data.frame(y, x_age, x_hrs_work, x_hrs_sleep)
synthetic_one <- synthesize(new, 1, n)
names(synthetic_one) <- c("OrigLogIncome", "SynLogIncome")
```

```
ggplot(synthetic_one, aes(x = OrigLogIncome, y = SynLogIncome)) +
  geom_point(size = 1) +
  labs(title = "Scatter plot of Synthetic log(Income) vs log(Income)") +
  theme_bw(base_size = 6, base_family = "")
```