IPUMS Health Data

MATH 301 Data Confidentiality

Henrik Olsson

February 25, 2020

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

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

```
##    AGE SEX RACEA EDUCREC2 HOURSWRK POORYN EARNIMP1 EARNIMPOINT1 USUALPL
## 1   37   1   100       54       40      9       70    11.855628       0
## 2   64   1   100       54       40      1       65    11.461632       2
## 4   53   2   100       54       40      1       32    10.819778       2
## 6   32   1   100       41       32      1        5    10.043249       0
## 9   52   1   100       60       45      1       68    11.592005       0
## 10  18   1   100       41        8      1        2     8.517193       2
##    DELAYCOST HINOTCOVE ALCDAYSWK CIGDAYMO HRSLEEP WORFREQ DEPFREQ
## 1          1         1        96       96       0       0       0
## 2          1         1        70       96       6       4       5
## 4          1         1        10       96       7       4       5
## 6          1         2        96       96       0       0       0
## 9          1         1        96       96       0       0       0
## 10         1         1        96       96      98       8       8
##       income
## 1  11.855628
## 2  11.461632
## 4  10.819778
## 6  10.043249
## 9  11.592005
## 10  8.517193
```

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 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.

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

```
## 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_hr_sleep[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$EARNIMPOINT1)
x_alc_one = as.vector(ipumsdata$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$.data_4)
x_hr_sleep = as.vector(ipumsdata$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,
"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_hr_sleep" = x_hr_sleep,
"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)
```

```
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", "beta8", "beta9", "beta10",
"beta11", "beta12", "beta13", "beta14", "sigma"),
adapt = 1000,
burnin = 5000,
sample = 5000,
thin = 10,
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 50000 iterations...
## Simulation complete
## Calculating summary statistics...
## Finished running the simulation
```

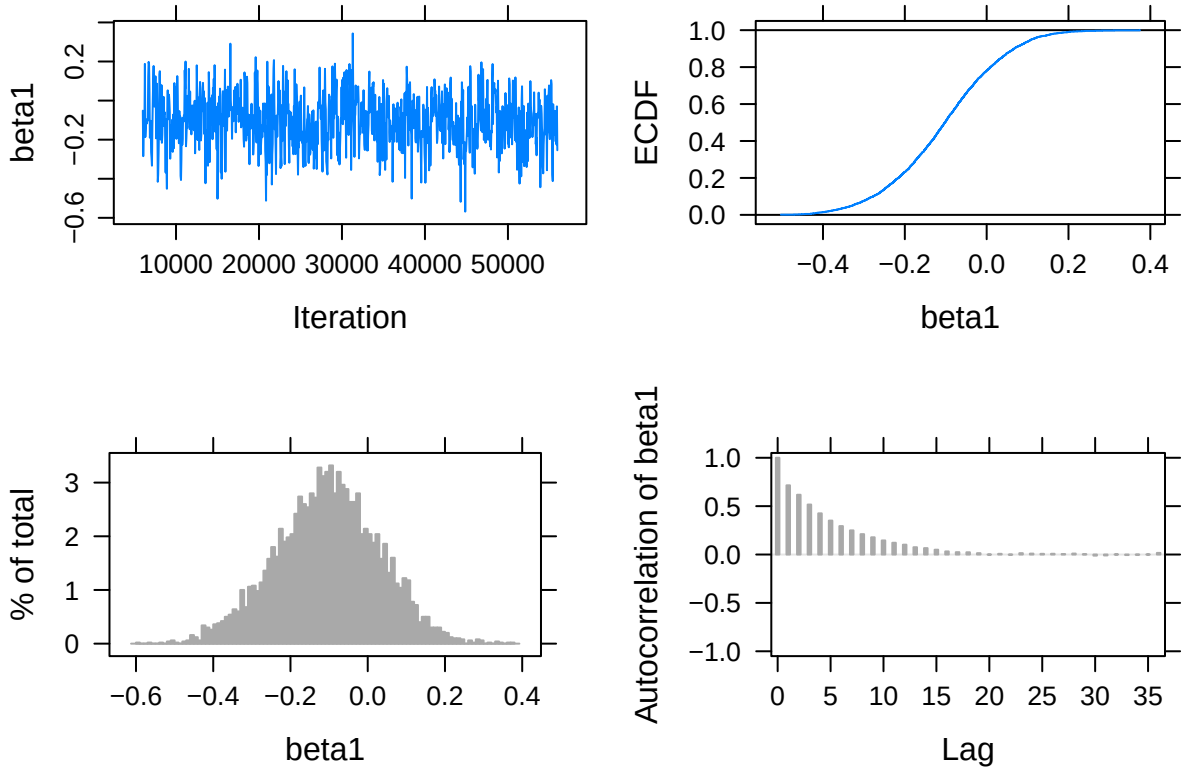
```
## JAGS output
summary(posterior_MLR)
```

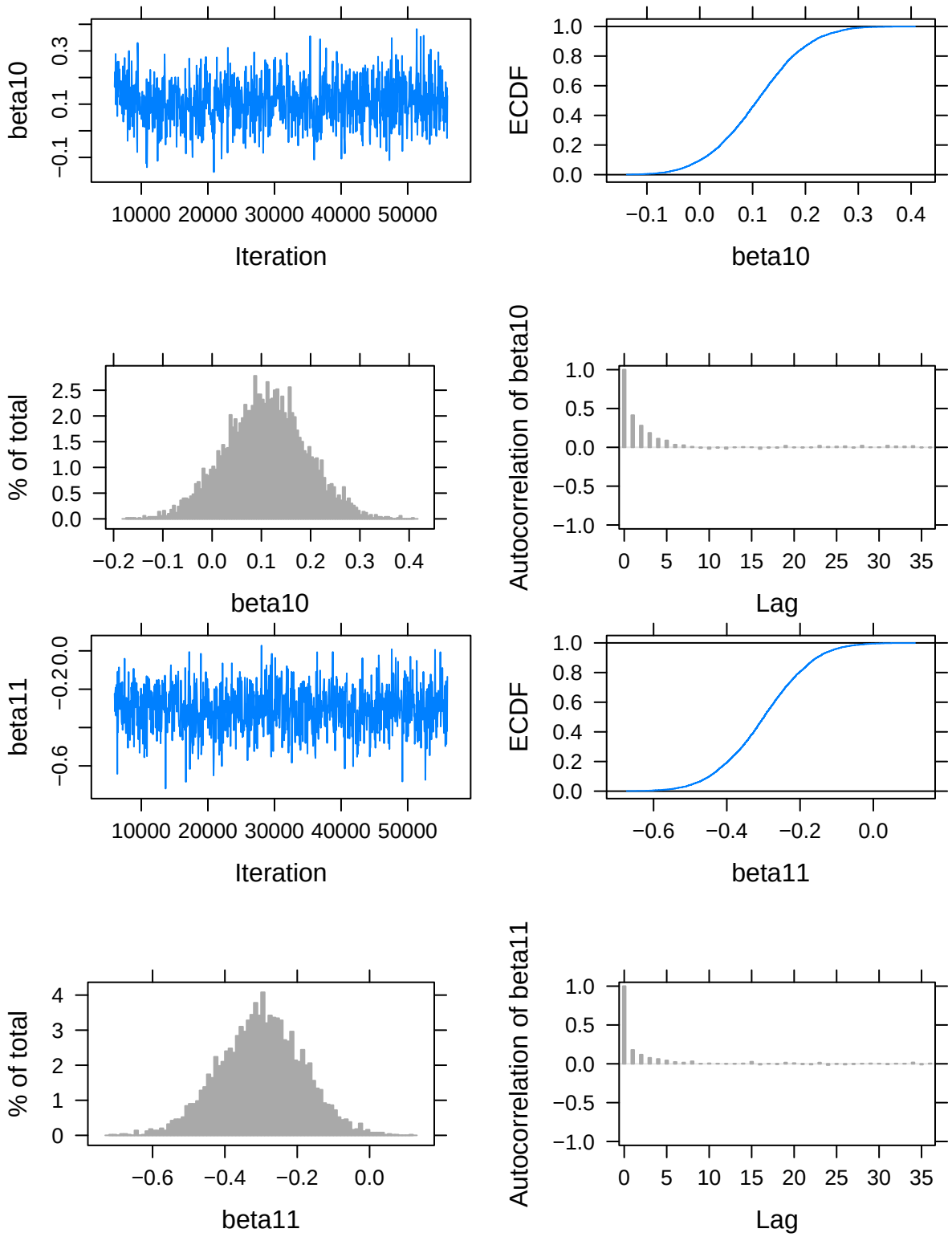
```
##             Lower95      Median     Upper95        Mean         SD Mode
## beta0   9.581394034  9.86026146 10.16311121  9.86140636 0.14970571   NA
## beta1  -0.366621203 -0.10206362  0.15722283 -0.10422771 0.13358079   NA
## beta2  -0.570208887 -0.27648774  0.01333201 -0.27763490 0.14831044   NA
## beta3  -0.755059217 -0.48773091 -0.18650212 -0.48747193 0.14388067   NA
## beta4  -0.146013778  0.12969343  0.41142850  0.12856125 0.14288045   NA
## beta5  -0.452498726 -0.11297559  0.24104787 -0.11671459 0.17695651   NA
## beta6  -0.009127746  0.31434407  0.63149381  0.31197786 0.16372070   NA
## beta7  -0.045679575  0.37952375  0.79288934  0.37771814 0.21318909   NA
```

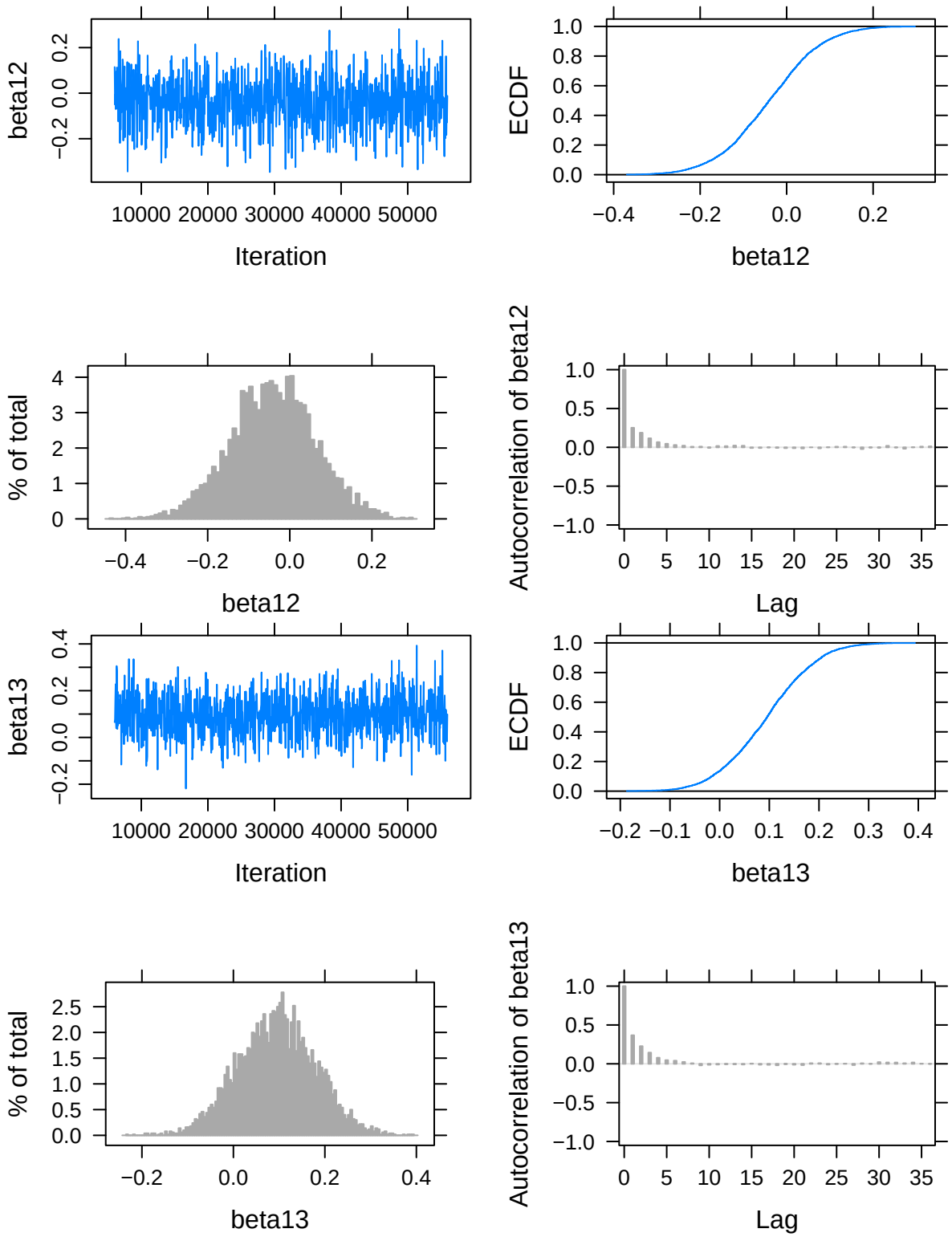
```
## beta8  -0.183054355  0.11866281  0.40522131  0.11808633 0.14892306   NA
## beta9   0.171661960  0.37614050  0.55861376  0.37506386 0.09899281   NA
## beta10 -0.057312859  0.10885814  0.26649764  0.10826492 0.08243687   NA
## beta11 -0.534791960 -0.29873582 -0.08495993 -0.30002936 0.11509112   NA
## beta12 -0.243317751 -0.04066058  0.16676727 -0.04085175 0.10370176   NA
## beta13 -0.074335055  0.09553502  0.25736813  0.09435571 0.08537747   NA
## beta14  0.535450662  0.64111573  0.73790209  0.64146801 0.05216169   NA
## sigma   1.189641841  1.21220688  1.23598651  1.21232248 0.01194880   NA
##               MCerr MC%ofSD SSeff       AC.100 psrf
## beta0  0.0075382957     5.0   394  0.184532436   NA
## beta1  0.0059779503     4.5   499  0.145440645   NA
## beta2  0.0060178696     4.1   607  0.119184685   NA
## beta3  0.0065503126     4.6   482  0.135846446   NA
## beta4  0.0059530026     4.2   576  0.131610613   NA
## beta5  0.0060223404     3.4   863  0.103881419   NA
## beta6  0.0060034161     3.7   744  0.086363291   NA
## beta7  0.0060615323     2.8  1237  0.066422929   NA
## beta8  0.0059459000     4.0   627  0.122199694   NA
## beta9  0.0029583846     3.0  1120 -0.034887457   NA
## beta10 0.0021261652     2.6  1503 -0.019614765   NA
## beta11 0.0023043108     2.0  2495  0.006224596   NA
## beta12 0.0022703066     2.2  2086 -0.005603353   NA
## beta13 0.0020479266     2.4  1738 -0.012318351   NA
## beta14 0.0009891535     1.9  2781 -0.012923582   NA
## sigma  0.0001689815     1.4  5000 -0.007506165   NA
```

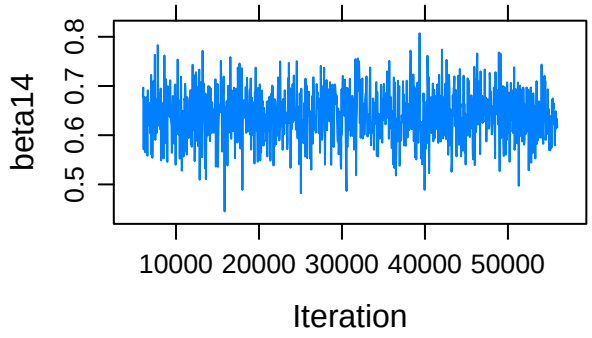
```
plot(posterior_MLR, vars = "beta1")
```

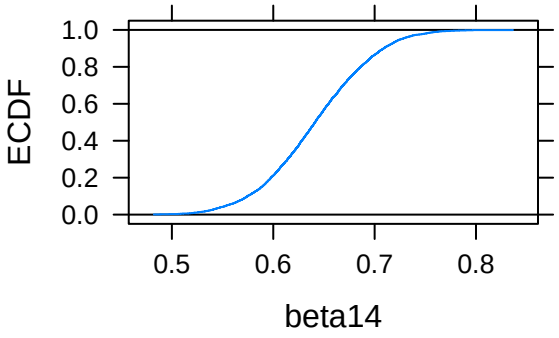
```
## Generating plots...
```

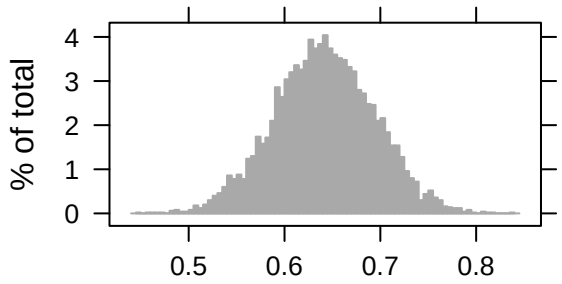


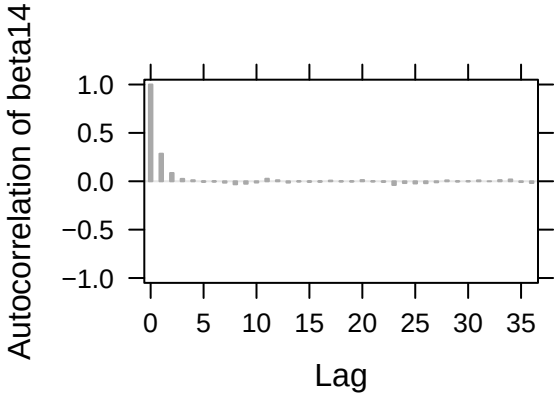

beta10
−0.1
0.1
0.3
10000 20000 30000 40000 50000
Iteration
ECDF
0.0
0.2
0.4
0.6
0.8
1.0
−0.1 0.0 0.1 0.2 0.3 0.4
beta10
% of total
0.0
0.5
1.0
1.5
2.0
2.5
−0.2 −0.1 0.0 0.1 0.2 0.3 0.4
beta10
Autocorrelation of beta10
−1.0
−0.5
0.0
0.5
1.0
0 5 10 15 20 25 30 35
Lag
beta11
−0.6
−0.2
0.0
10000 20000 30000 40000 50000
Iteration
ECDF
−0.6 −0.4 −0.2 0.0
beta11
% of total
0
1
2
3
4
−0.6 −0.4 −0.2 0.0
beta11
Autocorrelation of beta11
0 5 10 15 20 25 30 35
Lag

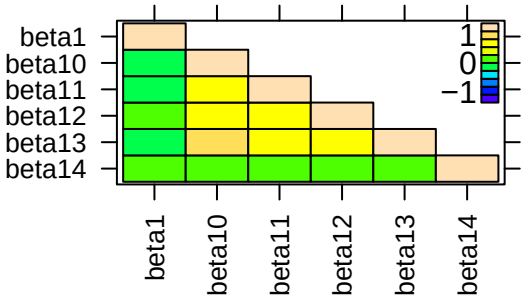

beta12
−0.2 0.0 0.2
10000 20000 30000 40000 50000
Iteration
ECDF
1.0
0.8
0.6
0.4
0.2
0.0
−0.4 −0.2 0.0 0.2
beta12
% of total
4
3
2
1
0
−0.4 −0.2 0.0 0.2
beta12
Autocorrelation of beta12
1.0
0.5
0.0
−0.5
−1.0
0 5 10 15 20 25 30 35
Lag
beta13
−0.2 0.0 0.2 0.4
10000 20000 30000 40000 50000
Iteration
ECDF
1.0
0.8
0.6
0.4
0.2
0.0
−0.2 −0.1 0.0 0.1 0.2 0.3 0.4
beta13
% of total
2.5
2.0
1.5
1.0
0.5
0.0
−0.2 0.0 0.2 0.4
beta13
Autocorrelation of beta13
1.0
0.5
0.0
−0.5
−1.0
0 5 10 15 20 25 30 35
Lag







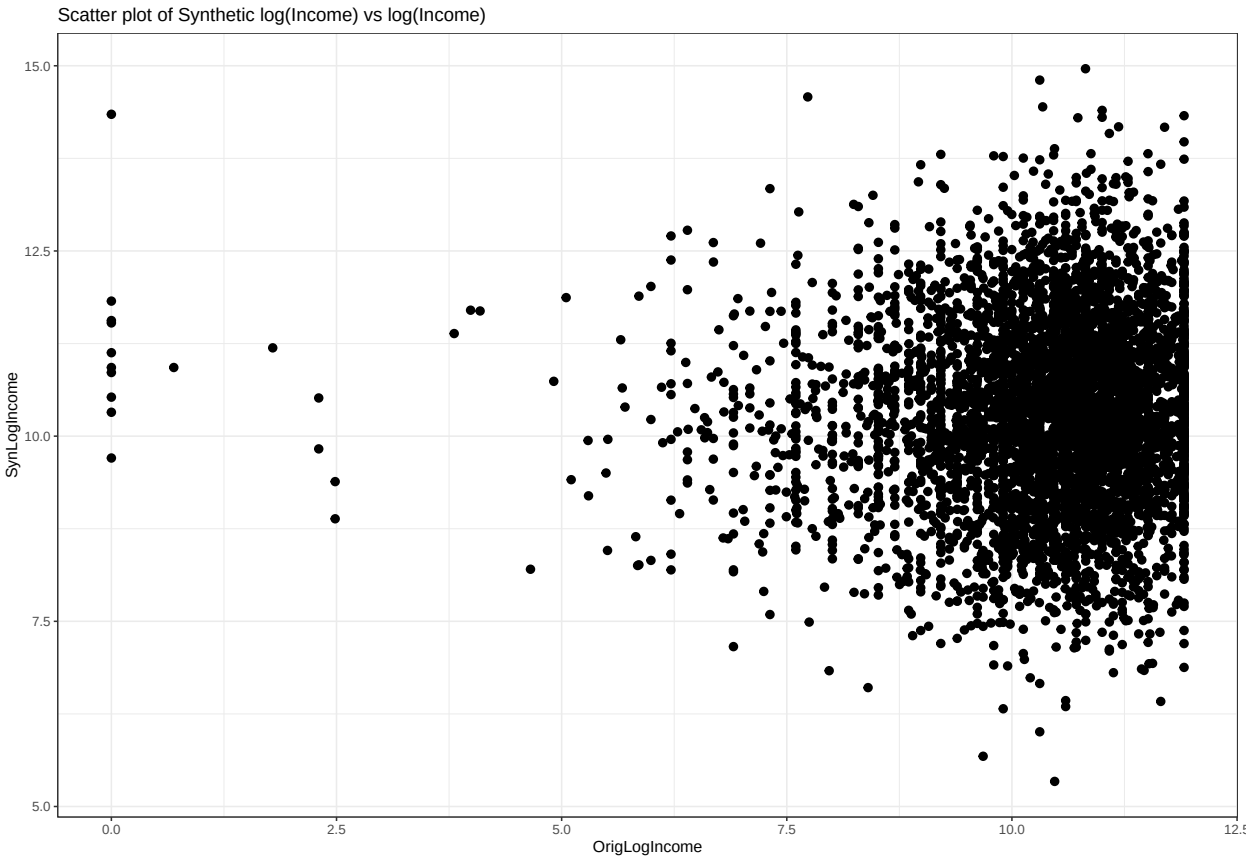





```
## Saving posterior parameter draws
post <- as.mcmc(posterior_MLR)

## 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 = "")
```


Scatter plot of Synthetic log(Income) vs log(Income)
SynLogIncome
OrigLogIncome
15.0
12.5
10.0
7.5
5.0
0.0
2.5
5.0
7.5
10.0