IPUMS Health Data

MATH 301 Data Confidentiality

Henrik Olsson

February 25, 2020

```
## read data
data <- read.csv("nhis_00001.csv")
## REMOVE THIS LATER. ONLY TAKING 1000 SAMPLES B/C JAGS TOOK TOO LONG
data <- sample_n(data, 2000, replace = FALSE, prob = NULL)
## log income
## data$LOGINC<- log(data$EARNIMPOINT1)
#load("tp.RData")
```

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.

```
## Remove all NIU (00) values
data <- data[!data$EDUCREC2 == 00, ]
data <- data[!data$HOURSWRK == 00, ]
data <- data[!data$HINOTCOVE == 0, ]
data <- data[!data$HRSLEEP == 00, ]
data <- data[!data$WORFREQ == 0, ]
```

```
## Create new column RACE and recode into 6 categories
## 1 = White, 2 = Black, 3 = American Indian, 4 = Asian,
## 5 = Other races, 6 = Two or more races
data <- data %>% mutate(RACE = ifelse(RACEA %in% 100, 1, ifelse(RACEA %in% 200, 2, ifelse(RACEA %in% c(3

## Create new column EDUC and recode into 3 categories
## 1 = 4 years of high school or less, 2 = 4 years of college,
## 3 = 5+ years of college
data <- data %>% mutate(EDUC = ifelse(EDUCREC2 %in% c(10,20,30,31,32,40,41,42), 1, ifelse(EDUCREC2 %in%

data <- data %>% mutate(INCOME = ifelse(EARNIMPOINT1 %in% 0, 0, 1))
```

```
head(data)
```

```
##   AGE SEX RACEA EDUCREC2 HOURSWRK POORYN EARNIMP1 EARNIMPOINT1 USUALPL
## 1  22   2   100       51       36      1        3        14000       2
## 2  32   2   200       54       35      1       21        37000       2
## 3  19   1   100       42       30      1        2         7000       2
## 4  66   1   200       60       40      1       52        70000       2
## 5  52   2   100       51       47      1       31        47000       2
## 6  20   2   100       42       15      9        0            0       1
##   DELAYCOST HINOTCOVE ALCDAYSWK CIGDAYMO HRSLEEP WORFREQ DEPFREQ RACE EDUC
## 1         2         1        80       22       6       1       2    1    2
## 2         1         1        96       96       7       5       5    2    2
## 3         1         1         0       96       8       5       4    1    1
## 4         1         1        80       96       8       5       5    2    3
## 5         1         1         0       96       6       2       4    1    2
## 6         2         2         0       96       6       2       2    1    1
##   INCOME
```

```
## 1      1
## 2      1
## 3      1
## 4      1
## 5      1
## 6      0
```

```
summary(data)
```

```
##       AGE              SEX            RACEA          EDUCREC2
##  Min.   :18.00   Min.   :1.00   Min.   :100.0   Min.   :10.00
##  1st Qu.:31.00   1st Qu.:1.00   1st Qu.:100.0   1st Qu.:42.00
##  Median :42.00   Median :1.00   Median :100.0   Median :51.00
##  Mean   :43.78   Mean   :1.49   Mean   :137.1   Mean   :50.67
##  3rd Qu.:56.00   3rd Qu.:2.00   3rd Qu.:100.0   3rd Qu.:54.00
##  Max.   :78.00   Max.   :2.00   Max.   :600.0   Max.   :60.00
##     HOURSWRK         POORYN         EARNIMP1      EARNIMPOINT1
##  Min.   : 1.00   Min.   :1.000   Min.   : 0.00   Min.   :     0
##  1st Qu.:35.25   1st Qu.:1.000   1st Qu.: 5.00   1st Qu.: 21451
##  Median :40.00   Median :1.000   Median :22.00   Median : 40000
##  Mean   :40.36   Mean   :1.315   Mean   :29.07   Mean   : 49986
##  3rd Qu.:50.00   3rd Qu.:1.000   3rd Qu.:52.00   3rd Qu.: 70000
##  Max.   :95.00   Max.   :9.000   Max.   :70.00   Max.   :149000
##     USUALPL        DELAYCOST       HINOTCOVE       ALCDAYSWK
##  Min.   :1.000   Min.   :1.000   Min.   :1.000   Min.   : 0.00
##  1st Qu.:2.000   1st Qu.:1.000   1st Qu.:1.000   1st Qu.: 0.00
##  Median :2.000   Median :1.000   Median :1.000   Median :10.00
##  Mean   :1.859   Mean   :1.126   Mean   :1.113   Mean   :32.85
##  3rd Qu.:2.000   3rd Qu.:1.000   3rd Qu.:1.000   3rd Qu.:70.00
##  Max.   :3.000   Max.   :2.000   Max.   :2.000   Max.   :99.00
##     CIGDAYMO        HRSLEEP          WORFREQ         DEPFREQ
##  Min.   : 4.00   Min.   : 3.000   Min.   :1.000   Min.   :1.000
##  1st Qu.:96.00   1st Qu.: 6.000   1st Qu.:3.000   1st Qu.:4.000
##  Median :96.00   Median : 7.000   Median :4.000   Median :5.000
##  Mean   :94.16   Mean   : 6.928   Mean   :3.677   Mean   :4.308
##  3rd Qu.:96.00   3rd Qu.: 8.000   3rd Qu.:5.000   3rd Qu.:5.000
##  Max.   :96.00   Max.   :12.000   Max.   :5.000   Max.   :5.000
##       RACE            EDUC           INCOME
##  Min.   :1.000   Min.   :1.000   Min.   :0.0000
##  1st Qu.:1.000   1st Qu.:1.000   1st Qu.:1.0000
##  Median :1.000   Median :2.000   Median :1.0000
##  Mean   :1.351   Mean   :1.905   Mean   :0.9744
##  3rd Qu.:1.000   3rd Qu.:2.000   3rd Qu.:1.0000
##  Max.   :5.000   Max.   :3.000   Max.   :1.0000
```

Step 1: Synthetic Logistic Regression Model

```
## JAGS script
modelString <-"
model {
## sampling
for(i in 1:N){
y[i] ~ dbern(p[i])
logit(p[i]) <- beta0 + beta1*x_age[i] +
```

```
beta2*x_sex_male[i] + beta3*x_sex_female[i] +
beta4*x_race_w[i] + beta5*x_race_b[i] +
beta6*x_race_i[i] + beta7*x_race_a[i] +
beta8*x_race_o[i] +
beta10*x_educ_1[i] + beta11*x_educ_2[i] +
beta12*x_educ_3[i] + beta13*x_hourswrk[i] +
beta14*x_health_cov[i] + beta15*x_health_nocov[i] +
beta16*x_hrsleep[i] + beta17*x_wor_daily[i] +
beta18*x_wor_weekly[i] + beta19*x_wor_monthly[i] +
beta20*x_wor_fewtimes[i] + beta21*x_wor_never[i]
}
## 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)
beta10 ~ dnorm(mu10, g10)
beta11 ~ dnorm(mu11, g11)
beta12 ~ dnorm(mu12, g12)
beta13 ~ dnorm(mu13, g13)
beta14 ~ dnorm(mu14, g14)
beta15 ~ dnorm(mu15, g15)
beta16 ~ dnorm(mu16, g16)
beta17 ~ dnorm(mu17, g17)
beta18 ~ dnorm(mu18, g18)
beta19 ~ dnorm(mu19, g19)
beta20 ~ dnorm(mu20, g20)
beta21 ~ dnorm(mu21, g21)
}"

#invsigma2 ~ dgamma(a, b)
#sigma <- sqrt(pow(invsigma2, -1))
```

```
y = as.vector(data$INCOME)
x_age = as.vector(data$AGE) ## age
x_sex_male = as.vector(data$SEX$.data_1) ## male
x_sex_female = as.vector(data$SEX$.data_2) ## female
x_race_w = as.vector(data$RACE$.data_1) ## white
x_race_b = as.vector(data$RACE$.data_2) ## black/african-american
x_race_i = as.vector(data$RACE$.data_3) ## american indian
x_race_a = as.vector(data$RACE$.data_4) ## asian
x_race_o = as.vector(data$RACE$.data_5) ## other races
x_educ_1 = as.vector(data$EDUC$.data_3) ## 4 years of high school or less
x_educ_2 = as.vector(data$EDUC$.data_1) ## 4 years of college
x_educ_3 = as.vector(data$EDUC$.data_2) ## 5+ years of college
x_hourswrk = as.vector(data$HOURSWRK) ## hours of work
x_health_cov = as.vector(data$HEALTH$.data_1) ## has health coverage
x_health_nocov = as.vector(data$HEALTH$.data_2) ## has no health coverage
x_hrsleep = as.vector(data$HRSLEEP) ## hours of sleep
```

```
x_wor_daily = as.vector(data$WORRY$.data_2) ## worry daily
x_wor_weekly = as.vector(data$WORRY$.data_5) ## worry weekly
x_wor_monthly = as.vector(data$WORRY$.data_4) ## worry monthly
x_wor_fewtimes = as.vector(data$WORRY$.data_3) ## worry few times a year
x_wor_never = as.vector(data$WORRY$.data_1) ## worry never

N = length(y) # Compute the number of observations

## Pass the data and hyperparameter values to JAGS
the_data <- list("y" = y,
"x_age" = x_age, "x_sex_male" = x_sex_male,
"x_sex_female" = x_sex_female, "x_race_w" = x_race_w,
"x_race_b" = x_race_b, "x_race_i" = x_race_i,
"x_race_a" = x_race_a, "x_race_o" = x_race_o,
"x_educ_1" = x_educ_1,
"x_educ_2" = x_educ_2, "x_educ_3" = x_educ_3,
"x_hourswrk" = x_hourswrk, "x_health_cov" = x_health_cov,
"x_health_nocov" = x_health_nocov, "x_hrsleep" = x_hrsleep,
"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,
"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,
"mu10" = 0, "g10" = 1, "mu11" = 0, "g11" = 1,
"mu12" = 0, "g12" = 1, "mu13" = 0, "g13" = 1,
"mu14" = 0, "g14" = 1, "mu15" = 0, "g15" = 1,
"mu16" = 0, "g16" = 1, "mu17" = 0, "g17" = 1,
"mu18" = 0, "g18" = 1, "mu19" = 0, "g19" = 1,
"mu20" = 0, "g20" = 1, "mu21" = 0, "g21" = 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", "beta10",
"beta11", "beta12", "beta13", "beta14", "beta15", "beta16", "beta17",
"beta18", "beta19", "beta20", "beta21"),
adapt = 1000,
burnin = 5000,
sample = 5000,
```

```
thin = 1,
inits = initsfunction)
```

```
## Loading required namespace: rjags
```

```
## Compiling rjags model...
## Calling the simulation using the rjags method...
## Adapting the model for 1000 iterations...
## 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
```

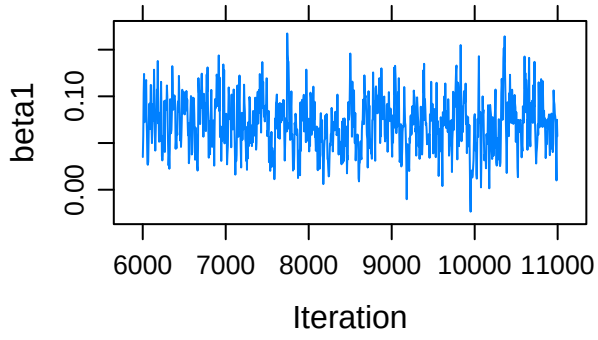
```
## JAGS output
summary(posterior_MLR)
```

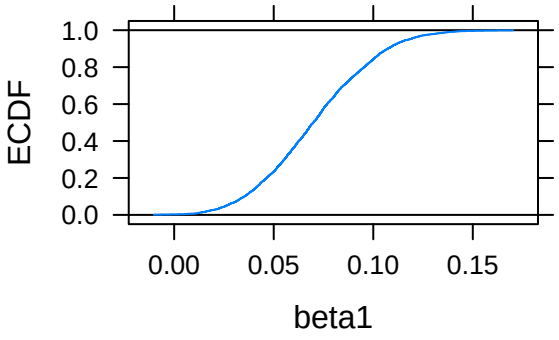
```
##            Lower95       Median   Upper95         Mean         SD Mode
## beta0  -1.94216286 -0.064370798 1.7087691 -0.059244782 0.95755888   NA
## beta1   0.01703532  0.069855437 0.1243447  0.070712076 0.02800400   NA
## beta2  -1.52081205 -0.010785325 1.5617397 -0.001100925 0.78853927   NA
## beta3  -1.42833200 -0.030058490 1.6036287 -0.037891187 0.76633484   NA
## beta4  -1.70854020 -0.238894845 1.2627460 -0.261122309 0.77038375   NA
## beta5  -1.90242561 -0.266598173 1.3632391 -0.247070020 0.83764811   NA
## beta6  -1.80407664  0.125012061 1.9626125  0.118639226 0.98157860   NA
## beta7  -1.46997316  0.258248644 2.0954143  0.279991684 0.91151560   NA
## beta8  -1.82410712  0.082366429 1.9653833  0.082791730 0.96465078   NA
## beta10 -1.80561893 -0.127869671 1.5390520 -0.129151490 0.86019312   NA
## beta11 -1.47775601  0.068138162 1.5119147  0.069102320 0.76951951   NA
## beta12 -1.46719304  0.042431748 1.5549516  0.024320356 0.77475482   NA
## beta13  0.04660710  0.100308705 0.1553877  0.101392498 0.02817009   NA
## beta14 -0.91600088  0.532118514 2.0267029  0.551548048 0.77527016   NA
## beta15 -2.20148535 -0.584046364 0.9382540 -0.598551876 0.79961200   NA
## beta16 -0.67530542 -0.231626161 0.2103327 -0.229364201 0.23163627   NA
## beta17 -1.74548598 -0.333244751 1.0957288 -0.331396086 0.72269332   NA
## beta18 -1.65502601 -0.169343372 1.1056249 -0.169482359 0.70711899   NA
## beta19 -0.56718725  0.987889468 2.5936241  0.994046300 0.81821663   NA
## beta20 -1.62605643  0.004998299 1.5255489  0.013384858 0.80716359   NA
## beta21 -1.99993607 -0.575112387 1.0555822 -0.548693811 0.77444022   NA
##              MCerr MC%ofSD SSeff         AC.10 psrf
## beta0  0.056190074     5.9   290  2.982882e-01   NA
## beta1  0.001626187     5.8   297  3.489078e-01   NA
## beta2  0.030278286     3.8   678  6.002516e-02   NA
## beta3  0.031597559     4.1   588  1.017022e-01   NA
## beta4  0.034138273     4.4   509  1.546571e-01   NA
## beta5  0.026542528     3.2   996  8.377229e-02   NA
## beta6  0.017770571     1.8  3051  2.837257e-02   NA
## beta7  0.018196566     2.0  2509  4.529739e-05   NA
## beta8  0.017802231     1.8  2936  6.935257e-03   NA
## beta10 0.019769596     2.3  1893  2.958121e-02   NA
## beta11 0.029331843     3.8   688  6.214165e-02   NA
## beta12 0.027965868     3.6   767  4.804411e-02   NA
## beta13 0.001304828     4.6   466  1.538657e-01   NA
```

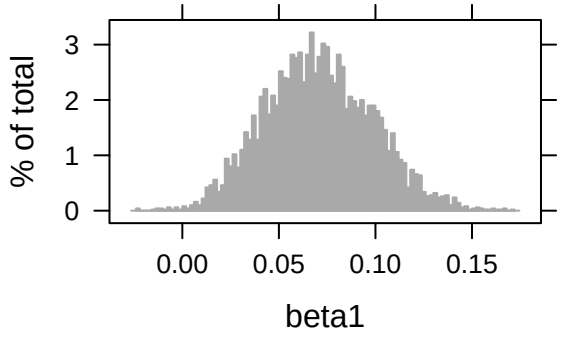
```
## beta14 0.033415330     4.3   538  1.097408e-01   NA
## beta15 0.028190107     3.5   805  5.718940e-02   NA
## beta16 0.022056978     9.5   110  6.568611e-01   NA
## beta17 0.018092380     2.5  1596  9.121764e-03   NA
## beta18 0.020274399     2.9  1216  1.647437e-02   NA
## beta19 0.016687399     2.0  2404 -2.150645e-02   NA
## beta20 0.016393582     2.0  2424 -4.418488e-02   NA
## beta21 0.019459208     2.5  1584  3.691483e-02   NA
```

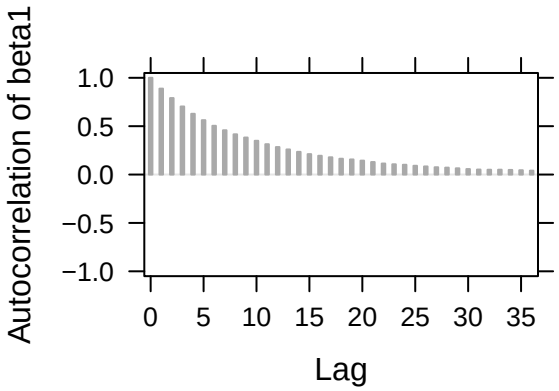
```
plot(posterior_MLR, vars = "beta1")
```

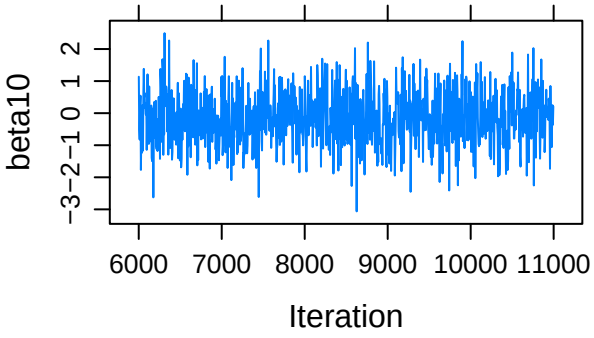
```
## Generating plots...
```

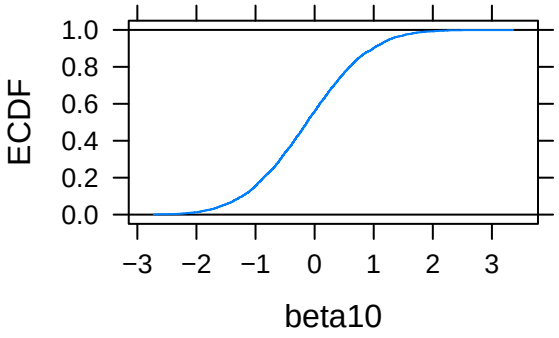


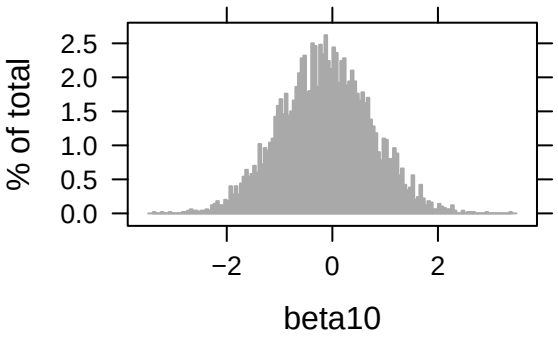


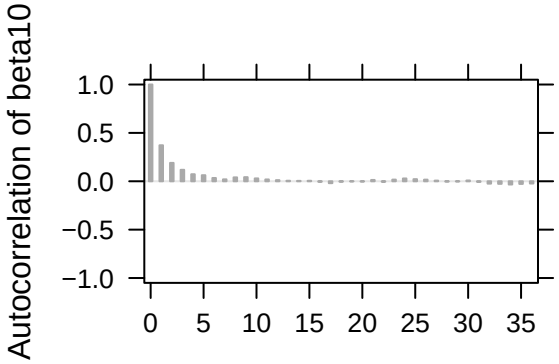


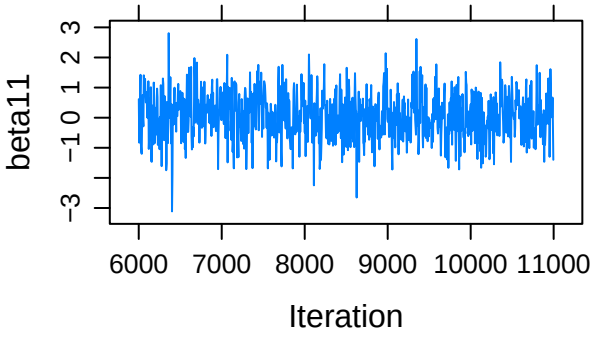


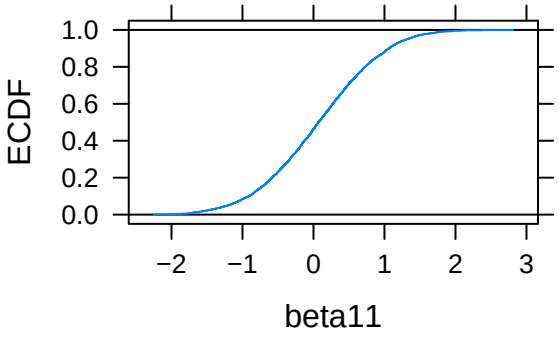

beta10
−3 −2 −1 0 1 2
6000 7000 8000 9000 10000 11000
Iteration

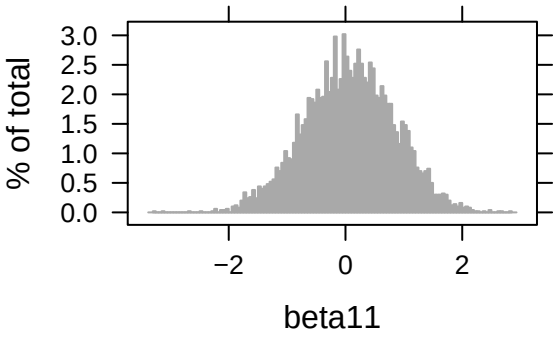

ECDF
0.0 0.2 0.4 0.6 0.8 1.0
−3 −2 −1 0 1 2 3
beta10

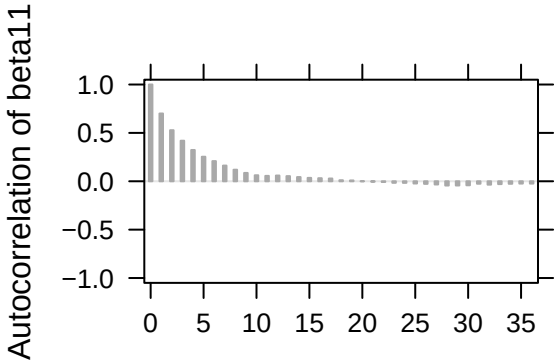

% of total
0.0 0.5 1.0 1.5 2.0 2.5
−2 0 2
beta10

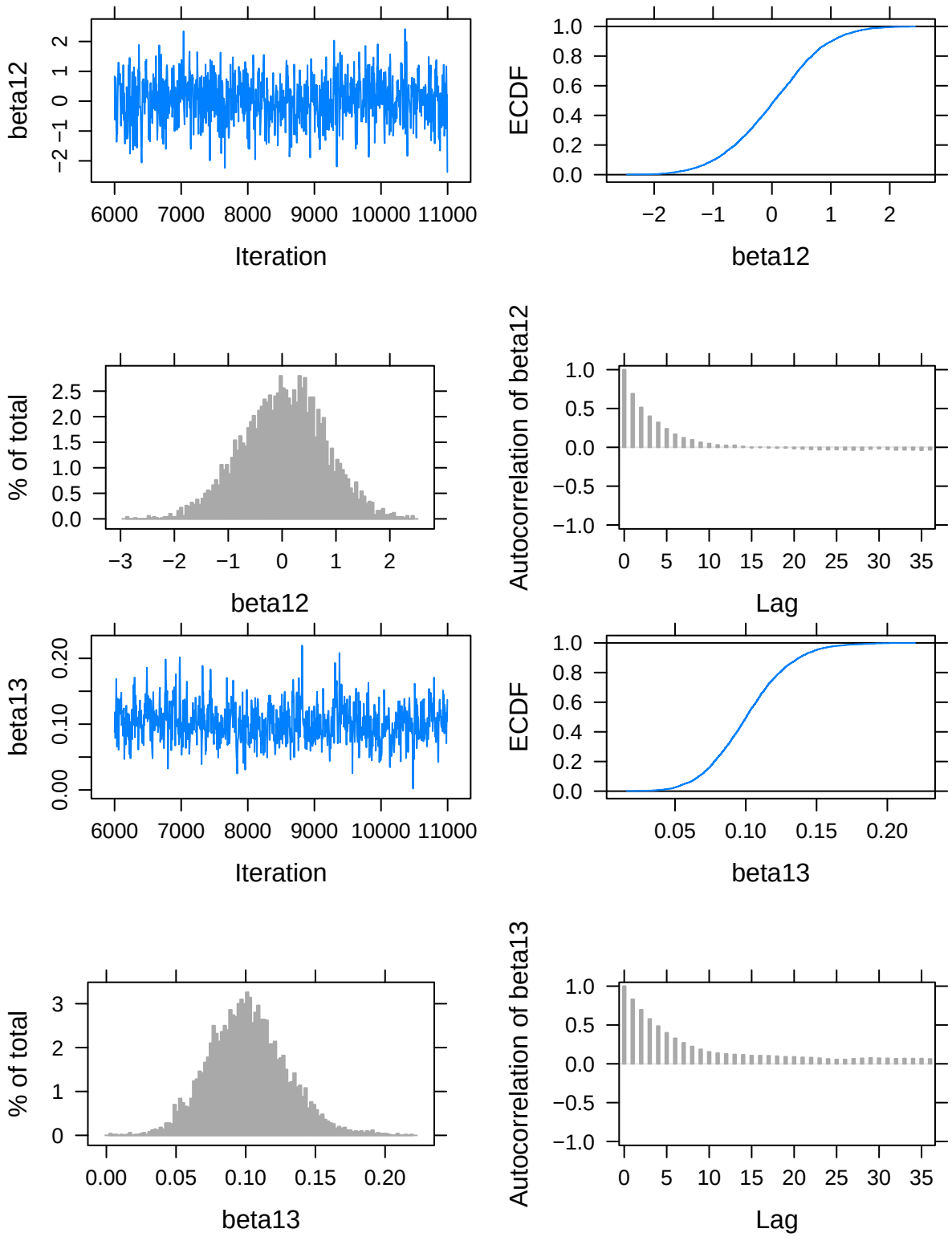

Autocorrelation of beta10
−1.0 −0.5 0.0 0.5 1.0
0 5 10 15 20 25 30 35
Lag

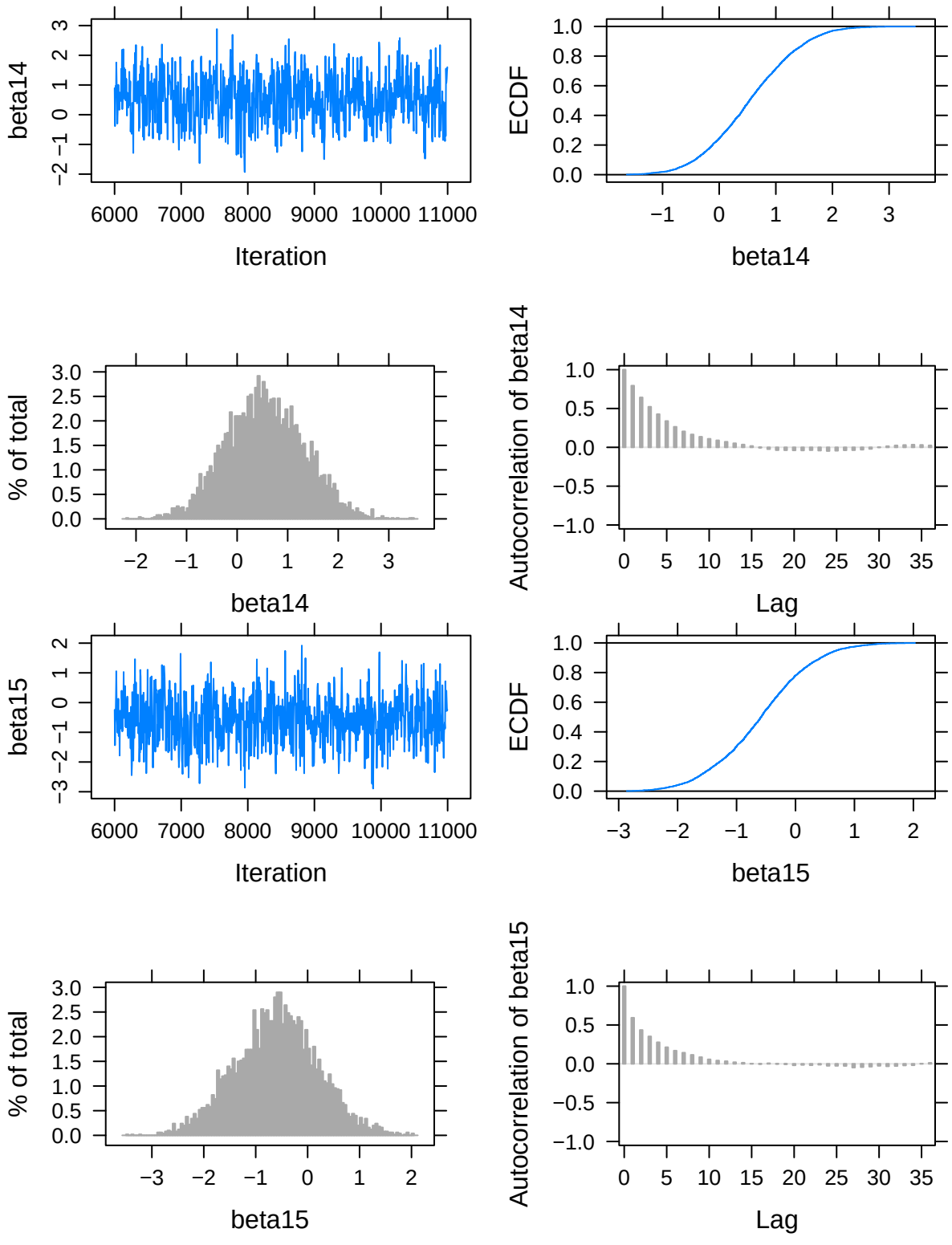

beta11
−3 −1 0 1 2 3
6000 7000 8000 9000 10000 11000
Iteration

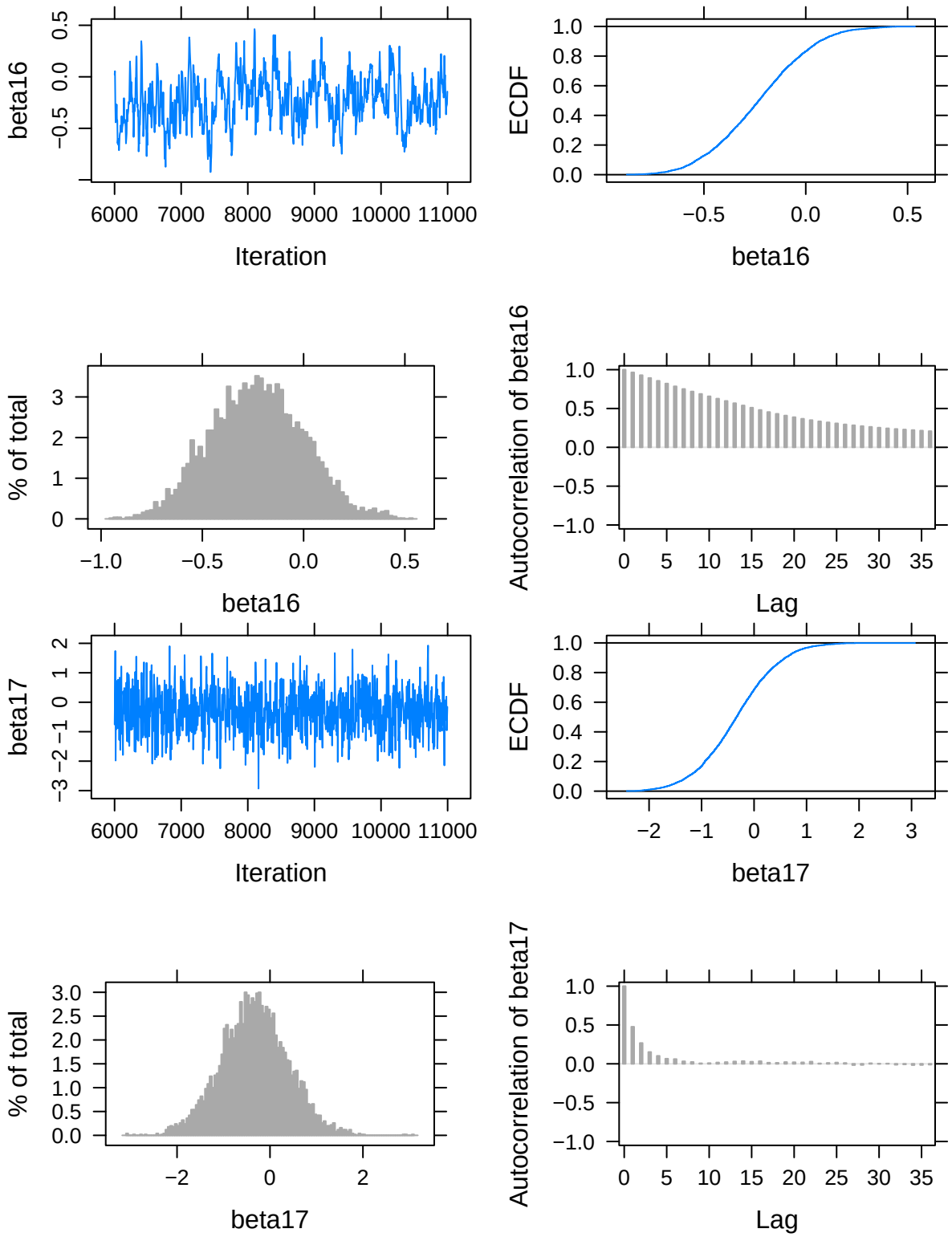

ECDF
0.0 0.2 0.4 0.6 0.8 1.0
−2 −1 0 1 2 3
beta11

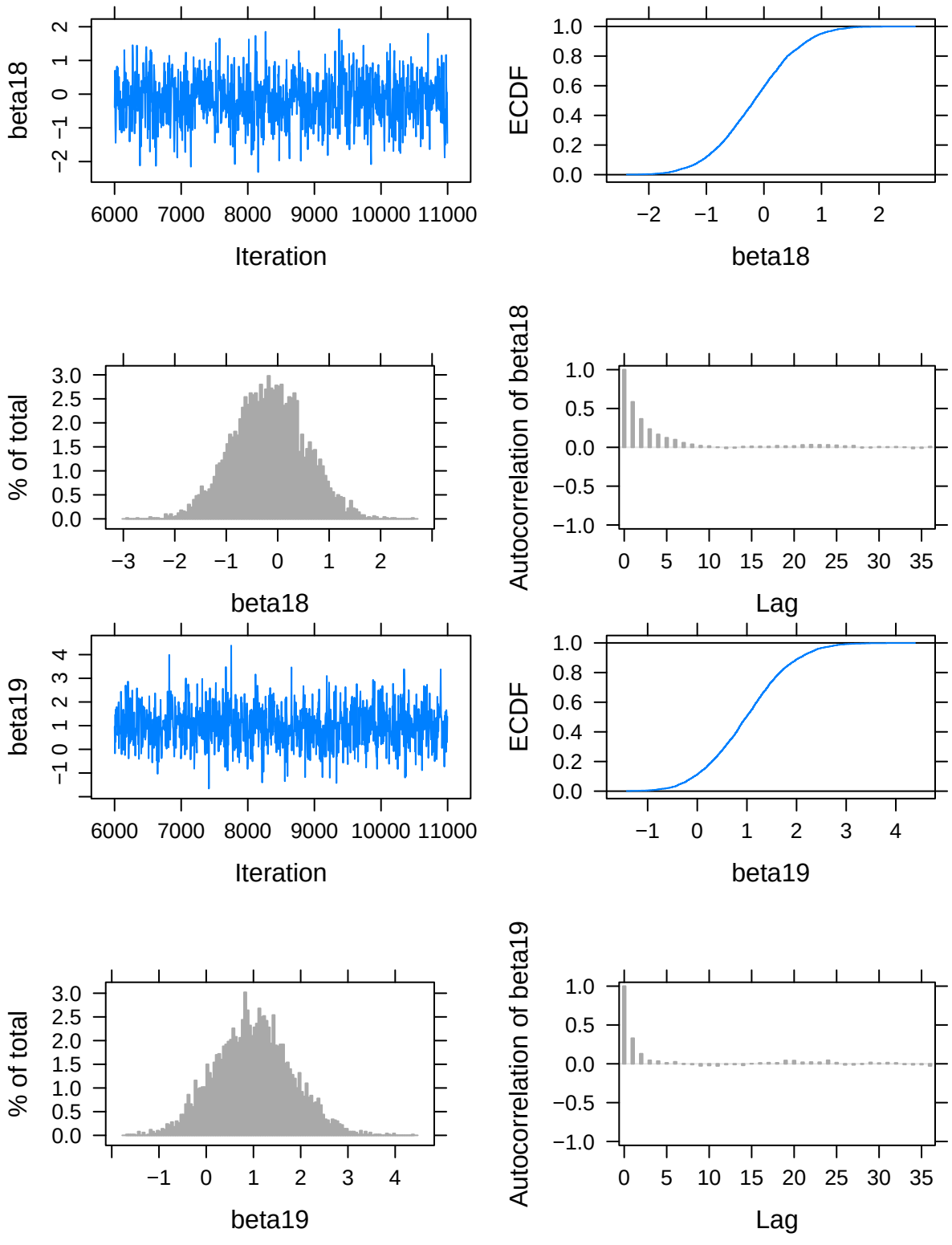

% of total
0.0 0.5 1.0 1.5 2.0 2.5 3.0
−2 0 2
beta11

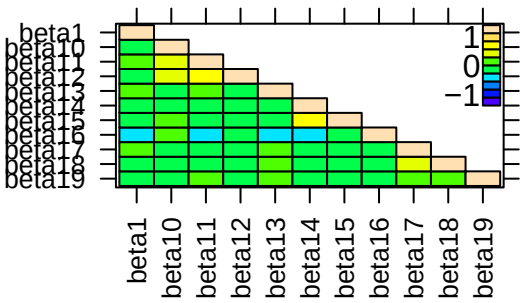

Autocorrelation of beta11
−1.0 −0.5 0.0 0.5 1.0
0 5 10 15 20 25 30 35
Lag


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
beta15
Iteration
ECDF
beta15
% of total
beta15
Autocorrelation of beta15
Lag


beta16
Iteration
ECDF
beta16
% of total
beta16
Autocorrelation of beta16
Lag
beta17
Iteration
ECDF
beta17
% of total
beta17
Autocorrelation of beta17
Lag


beta18
Iteration
ECDF
beta18
% of total
beta18
Autocorrelation of beta18
Lag
beta19
Iteration
ECDF
beta19
% of total
beta19
Autocorrelation of beta19
Lag




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

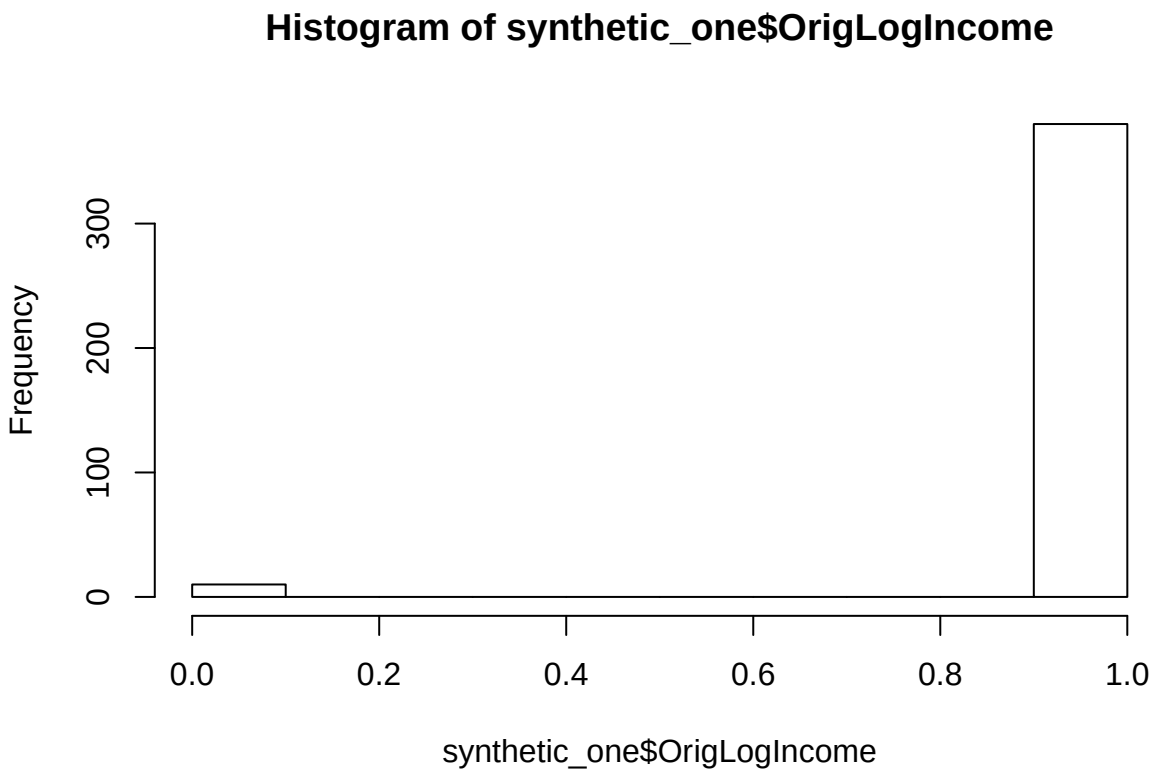
```
post[1, "beta0"]
```

```
## [1] 0.2047922
```

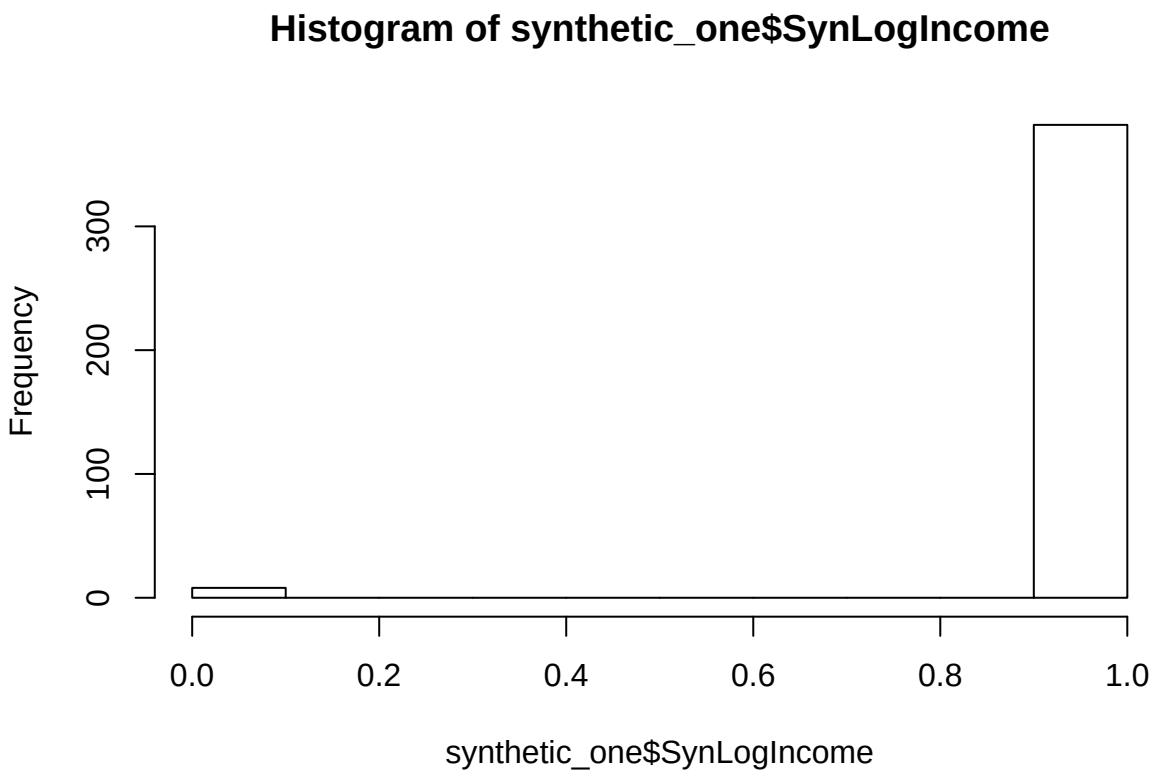
```
## Generating one set of sythetic data
synthesize <- function(X, index, n){
  synthetic_Y <- c()
  for(i in 1:n){
     p <- plogis(post[index, "beta0"] +  X$x_age[i] * post[index, "beta1"] +  X$x_sex_male[i] * post[inc
     synthetic_Y[i] <- rbinom(1,1,p)
  }
  data.frame(X$y, synthetic_Y)
}
n <- dim(data)[1]
params <- data.frame(y, x_age, x_sex_male, x_sex_female, x_race_w, x_race_b, x_race_i, x_race_a, x_race_

synthetic_one <- synthesize(params, 1, n)
names(synthetic_one) <- c("OrigLogIncome", "SynLogIncome")
```

```
hist(synthetic_one$OrigLogIncome)
```

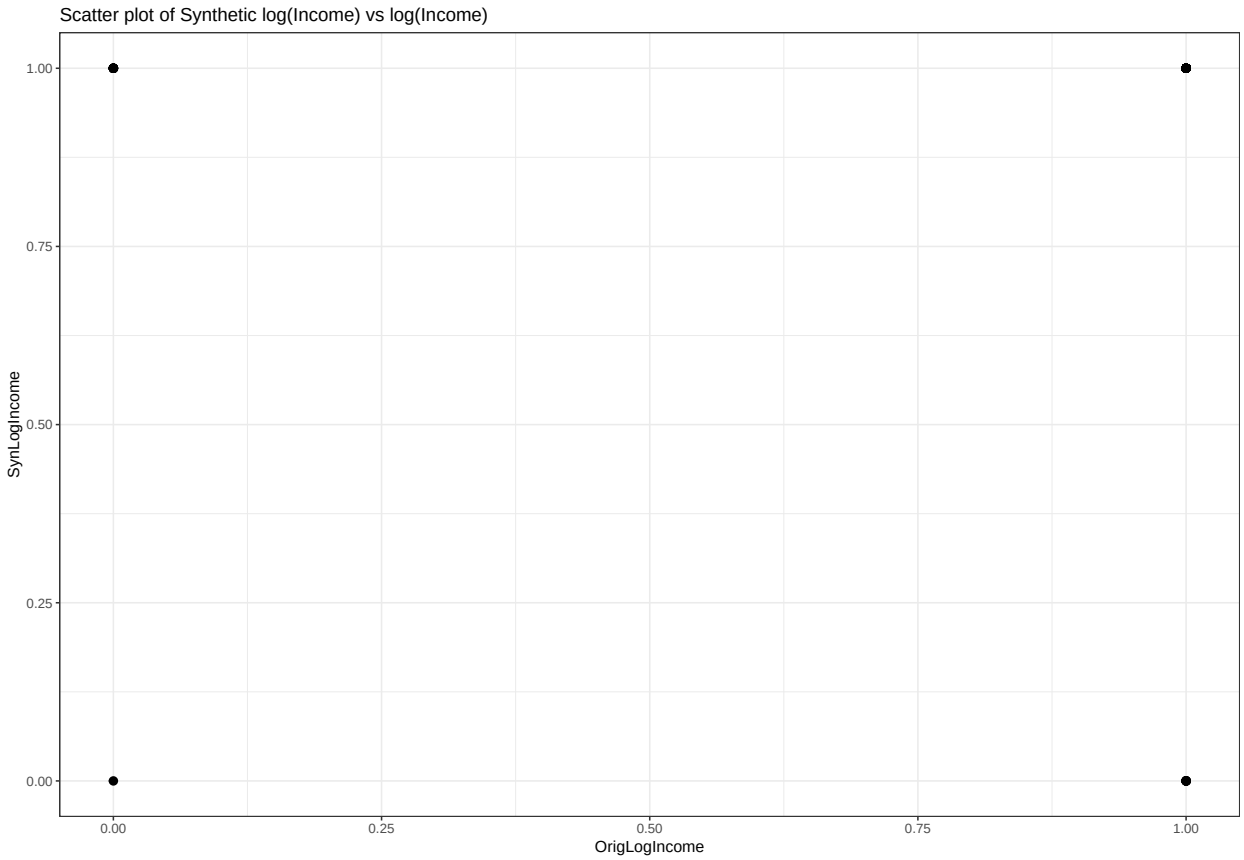



```r
hist(synthetic_one$SynLogIncome)
```




```r
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 = "")
```



Utility Evaluation

Analysis-specific measures

Synthetic IPUMS sample: step 1

```
n <- dim(data)[1]
m <- 20
synthetic_m <- vector("list",m)

for(i in 1:m){
  seed <- round(runif(1, 1, 1000))
  synthetic_new <- synthesize(params, 2000 + i, n)
  names(synthetic_new) <- c("params", "LogIncome")
  synthetic_m[[i]] <- synthetic_new
}
q <- rep(NA, m)
v <- rep(NA, m)
for (i in 1:m){
  synthetic_new <- synthetic_m[[i]]
  q[i] <- mean(synthetic_new$LogIncome)
  v[i] <- var(synthetic_new$LogIncome)/n
}
```

Synthetic IPUMS sample: step 2

```
q_bar_m <- mean(q)
b_m <- var(q)
v_bar_m <- mean(v)
```

Synthetic IPUMS sample: step 3

```
T_p <- b_m / m + v_bar_m
v_p <- (m - 1) * (1 + v_bar_m / (b_m / m))^2
```

Synthetic CE sample: step 4

- Obtain the point estimate for mean estimand Q, and the 95% confidence interval

```
set.seed(123)
q_bar_m
```

```
## [1] 0.9725641
```

```
t_score_syn <- qt(p = 0.975, df = v_p)
c(q_bar_m - t_score_syn * sqrt(T_p), q_bar_m + t_score_syn * sqrt(T_p))
```

```
## [1] 0.9557478 0.9893804
```

Synthetic mean estimand: 0.974 Synthetic 95% CI: [0.957, 0.990]

Synthetic CE sample: step 4-extra

```
## Obtain the point estimate for mean estimand Q, and the 95%
## confidence internval from the original data
set.seed(123)
mean_org <- mean(data$INCOME)
sd_org <- sd(data$INCOME)
t_score_org <- qt(p = 0.975, df = n-1)
mean_org
```

```
## [1] 0.974359
```

```
set.seed(123)
c(mean_org - t_score_org * sd_org / sqrt(n), mean_org + t_score_org * sd_org / sqrt(n))
```

```
## [1] 0.9586027 0.9901153
```

Original mean estimand: 0.970 Original 95% CI: [0.952, 0.987]

Interval overlap utility measure

Combining rules provide point estimate and confidence interval estimates. We can also obtain point estimate and confidence interval estimate from the original, confidential data.

```
L_s = quantile(synthetic_new$params, 0.025)
U_s = quantile(synthetic_new$params, 0.975)
L_o = quantile(data$INCOME, 0.025)
U_o = quantile(data$INCOME, 0.975)
L_i = max(L_s, L_o)
U_i = min(U_s, U_o)
I = (U_i - L_i) / (2 * (U_o - L_o)) + (U_i - L_i)/ (2 * (U_s - L_s))
I
```

```
## 97.5%
##     1
```

It seems like the interval overlap measure is 1 so we have high utility. We can potentially try to calculate multiple estimands and then take the average values of I over all estimands to obtain a summary.

Identification Risk Evaluation

First, we look into identification disclosure through expected match risk, true match rate, and false match rate. Then attribute disclosure will be examined.

Higher expected match risk, higher true match rate, and lower false match rate indicate higher identification disclosure risk for the sample. Since we are doing a two-part synthesis model, we could potentially calculate three summaries on each synthetic dataset, and take the average.

Step 1: calculate key quantities

CalculateKeyQuantities <- function(data, params, known.vars, syn.vars, n){ data <- data params <- params n <- n c_vector <- rep(NA, n) T_vector <- rep(NA, n) for (i in 1:n){ match <- (eval(parse(text=paste("data", *syn.vars*, "[*i*] == *params*",syn.vars,sep="",collapse="&")))& eval(parse(text=paste("data", *kn params*",known.vars,sep="",collapse="&")))) match.prob <- ifelse(match, 1/sum(match), 0)

```
if (max(match.prob) > 0){
  c_vector[i] <- length(match.prob[match.prob == max(match.prob)])
}
else
  c_vector[i] <- 0
  T_vector[i] <- is.element(i, rownames(data)[match.prob == max(match.prob)])
```

}

K_vector <- (c_vector * T_vector == 1) F_vector <- (c_vector * (1 - T_vector) == 1) s <- length(c_vector[c_vector == 1 & is.na(c_vector) == FALSE])

res_r <- list(c_vector = c_vector, T_vector = T_vector, K_vector = K_vector, F_vector = F_vector, s = s) return(res_r) }

Step 1: calculate key quantities cont'd

known.vars <- c("SEX", "RACE", "AGE", "EDUC","HOURSWRK", "HEALTH", "HRSLEEP","WORRY") syn.vars <- c("INCOME") n <- dim(data)[1] KeyQuantities <- CalculateKeyQuantities(data, param, known.vars, syn.vars, n)

Step 2: calculate 3 summary measures

IdentificationRisk <- function(c_vector, T_vector, K_vector, F_vector, s, N){

nonzero_c_index <- which(c_vector > 0) exp_match_risk <- sum(1/c_vector[nonzero_c_index]*T_vector[nonzero_c_index] true_match_rate <- sum(na.omit(K_vector))/N false_match_rate <- sum(na.omit(F_vector))/s res_r <- list(exp_match_risk = exp_match_risk, true_match_rate = true_match_rate, false_match_rate = false_match_rate) return(res_r) }

Step 2: calculate 3 summary measures cont'd

- each record is a target, therefore `N = n`

c_vector <- KeyQuantities[["c_vector"]] T_vector <- KeyQuantities[["T_vector"]] K_vector <- KeyQuantities[["K_vector"]] F_vector <- KeyQuantities[["F_vector"]] s <- KeyQuantities[["s"]] N <- n ThreeSummaries <- IdentificationRisk(c_vector, T_vector, K_vector, F_vector, s, N)

Step 2: calculate 3 summary measures cont'd

ThreeSummaries[["exp_match_risk"]] ThreeSummaries[["true_match_rate"]] ThreeSummaries[["false_match_rate"]]