IPUMS Health Data

MATH 301 Data Confidentiality

Henrik Olsson and Kevin Ros

April 19, 2020

```
## read data
#data <- read.csv("fulldataset.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)
#save(data,file="tp.RData")
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(
## 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   1   100       51       52      1       21        35000       2
## 2  55   1   200       31       30      2        4        18000       2
## 3  33   2   200       54       20      1        0            0       1
## 4  54   2   100       51       40      1        4        17500       2
## 5  66   2   100       51       20      1       12        32000       2
## 6  28   2   100       60       40      1       51        68000       2
##   DELAYCOST HINOTCOVE ALCDAYSWK CIGDAYMO HRSLEEP WORFREQ DEPFREQ RACE EDUC
## 1         2         1        10       96       8       2       2    1    2
## 2         1         2        96       96       6       5       5    2    1
## 3         1         1        96       96       6       4       4    2    2
## 4         1         1        96       96       4       4       5    1    2
## 5         1         1         0       96       9       4       5    1    2
## 6         1         1         0       96       8       2       5    1    3
##   INCOME
## 1      1
```

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

```r
summary(data)
```

```
##       AGE             SEX            RACEA        EDUCREC2
##  Min.   :18.00   Min.   :1.000   Min.   :100   Min.   :20.00
##  1st Qu.:31.00   1st Qu.:1.000   1st Qu.:100   1st Qu.:48.75
##  Median :44.00   Median :2.000   Median :100   Median :51.00
##  Mean   :44.63   Mean   :1.514   Mean   :133   Mean   :51.00
##  3rd Qu.:56.00   3rd Qu.:2.000   3rd Qu.:100   3rd Qu.:54.00
##  Max.   :84.00   Max.   :2.000   Max.   :600   Max.   :60.00
##     HOURSWRK        POORYN         EARNIMP1      EARNIMPOINT1
##  Min.   : 1.00   Min.   :1.000   Min.   : 0.00   Min.   :     0
##  1st Qu.:36.00   1st Qu.:1.000   1st Qu.: 5.00   1st Qu.: 20000
##  Median :40.00   Median :1.000   Median :22.00   Median : 40000
##  Mean   :40.36   Mean   :1.363   Mean   :29.45   Mean   : 50537
##  3rd Qu.:45.00   3rd Qu.:1.000   3rd Qu.:54.25   3rd Qu.: 72750
##  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 :20.00
##  Mean   :1.901   Mean   :1.157   Mean   :1.085   Mean   :30.52
##  3rd Qu.:2.000   3rd Qu.:1.000   3rd Qu.:1.000   3rd Qu.:70.00
##  Max.   :3.000   Max.   :9.000   Max.   :2.000   Max.   :99.00
##     CIGDAYMO        HRSLEEP          WORFREQ          DEPFREQ
##  Min.   : 1.00   Min.   : 3.000   Min.   :1.000   Min.   :1.00
##  1st Qu.:96.00   1st Qu.: 6.000   1st Qu.:3.000   1st Qu.:4.00
##  Median :96.00   Median : 7.000   Median :4.000   Median :5.00
##  Mean   :93.74   Mean   : 6.945   Mean   :3.703   Mean   :4.39
##  3rd Qu.:96.00   3rd Qu.: 8.000   3rd Qu.:5.000   3rd Qu.:5.00
##  Max.   :96.00   Max.   :16.000   Max.   :5.000   Max.   :9.00
##       RACE            EDUC           INCOME
##  Min.   :1.000   Min.   :1.000   Min.   :0.0000
##  1st Qu.:1.000   1st Qu.:1.750   1st Qu.:1.0000
##  Median :1.000   Median :2.000   Median :1.0000
##  Mean   :1.313   Mean   :1.931   Mean   :0.9698
##  3rd Qu.:1.000   3rd Qu.:2.000   3rd Qu.:1.0000
##  Max.   :5.000   Max.   :3.000   Max.   :1.0000
```

Part 1: Synthetic Logistic Regression Model (syn income into 0 or non-zero)

```r
## JAGS script
modelString_part1 <-"
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)
}"
```

```
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
```

```r
x_wor_never = as.vector(data$WORRY$.data_1) ## worry never
N = length(y) # Compute the number of observations
```

```r
## Pass the data and hyperparameter values to JAGS
the_data_part1 <- 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)
```

```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))
}
## Run the JAGS code for this model:
posterior_MLR <- run.jags(modelString_part1,
n.chains = 1,
data = the_data_part1,
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
```

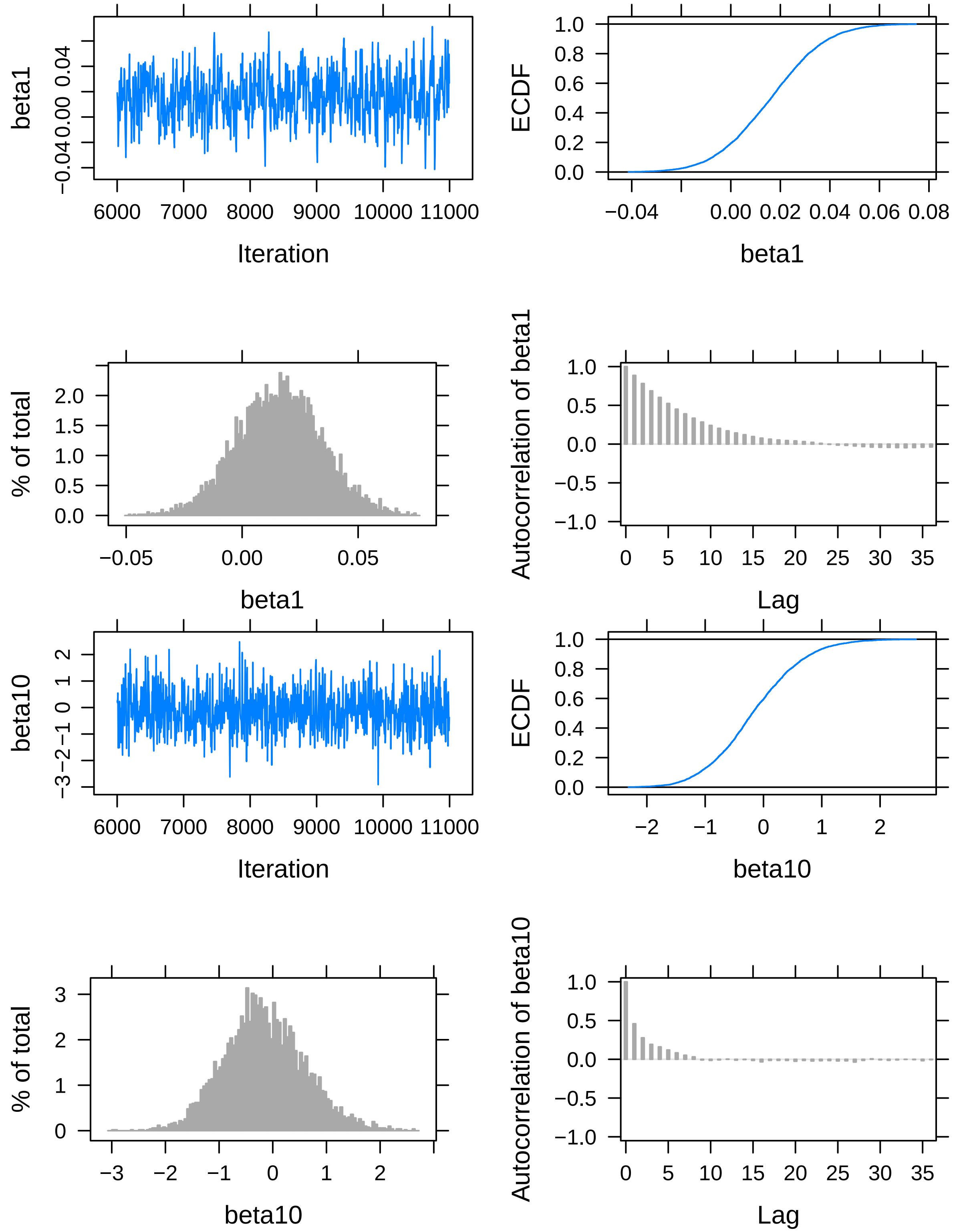
```
## JAGS output
summary(posterior_MLR)
```

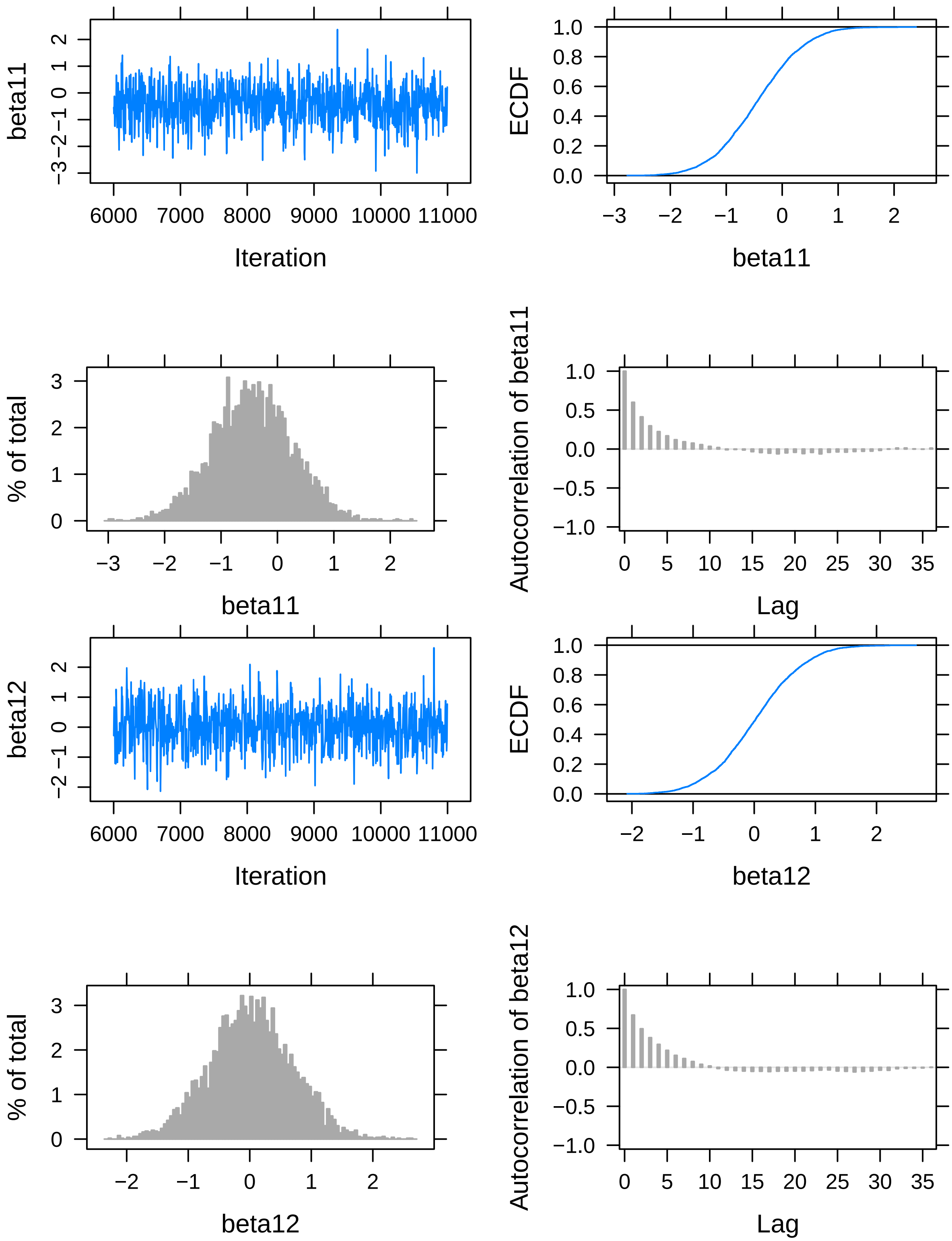
```
##              Lower95      Median    Upper95        Mean         SD Mode
## beta0  -2.300856118 -0.57504703 1.39462718 -0.55184910 0.96836675   NA
## beta1  -0.020483592  0.01622164 0.05214086  0.01606626 0.01848465   NA
## beta2  -1.756593676 -0.27861797 1.21263406 -0.26531145 0.76121138   NA
## beta3  -1.504832854 -0.16823668 1.35232090 -0.15472647 0.75527463   NA
## beta4  -1.322965361  0.07471915 1.44795764  0.07119505 0.69726326   NA
## beta5  -1.502777964 -0.02972413 1.71138614 -0.02165509 0.80690054   NA
## beta6  -1.704654040  0.14578204 1.93348460  0.15136173 0.92059642   NA
## beta7  -2.612923286 -1.01793851 0.60726307 -1.00743524 0.82953245   NA
## beta8  -1.897319248 -0.03249065 2.02531966 -0.00336818 0.99993317   NA
## beta10 -1.600578493 -0.19366971 1.31300710 -0.15458388 0.75188341   NA
## beta11 -1.865902174 -0.44435448 0.87274453 -0.43998727 0.70907848   NA
## beta12 -1.306321070  0.02025687 1.32639248  0.02271391 0.67985263   NA
## beta13  0.025260075  0.07170888 0.11859476  0.07208139 0.02355503   NA
## beta14 -2.741300442 -1.01792075 0.68759418 -1.00984107 0.88797838   NA
## beta15 -1.315658135  0.44960949 2.16954463  0.44510705 0.88198018   NA
## beta16 -0.006565879  0.39937857 0.75978322  0.39889405 0.19680198   NA
## beta17 -1.166421817  0.22402653 1.74113755  0.23419313 0.74200097   NA
## beta18 -0.978382945  0.32727673 1.70499268  0.34015632 0.69968194   NA
## beta19 -1.357169321 -0.06281715 1.29199769 -0.06301348 0.67830412   NA
## beta20 -2.583734071 -1.24523311 0.16184460 -1.23475412 0.69925180   NA
## beta21 -1.392484999  0.15007500 1.65319420  0.14201303 0.78051428   NA
##              MCerr MC%ofSD SSeff         AC.10 psrf
## beta0  0.065858926     6.8   216  0.393495829   NA
## beta1  0.001065753     5.8   301  0.241135459   NA
## beta2  0.033245033     4.4   524  0.162793870   NA
## beta3  0.033154448     4.4   519  0.149067669   NA
## beta4  0.030672827     4.4   517  0.083075970   NA
## beta5  0.017490042     2.2  2128  0.027911123   NA
## beta6  0.016667425     1.8  3051 -0.003835512   NA
## beta7  0.020503526     2.5  1637  0.052722482   NA
## beta8  0.017688602     1.8  3196 -0.017699675   NA
## beta10 0.021000935     2.8  1282 -0.014125249   NA
## beta11 0.020178703     2.8  1235  0.034809482   NA
## beta12 0.024462410     3.6   772  0.017694137   NA
## beta13 0.001261034     5.4   349  0.214915127   NA
## beta14 0.053128020     6.0   279  0.339794473   NA
## beta15 0.019256825     2.2  2098  0.034306459   NA
## beta16 0.017335878     8.8   129  0.590667751   NA
## beta17 0.016471518     2.2  2029  0.017948160   NA
## beta18 0.018240837     2.6  1471  0.010314591   NA
## beta19 0.017980577     2.7  1423 -0.013357879   NA
## beta20 0.018671192     2.7  1403  0.007185830   NA
```

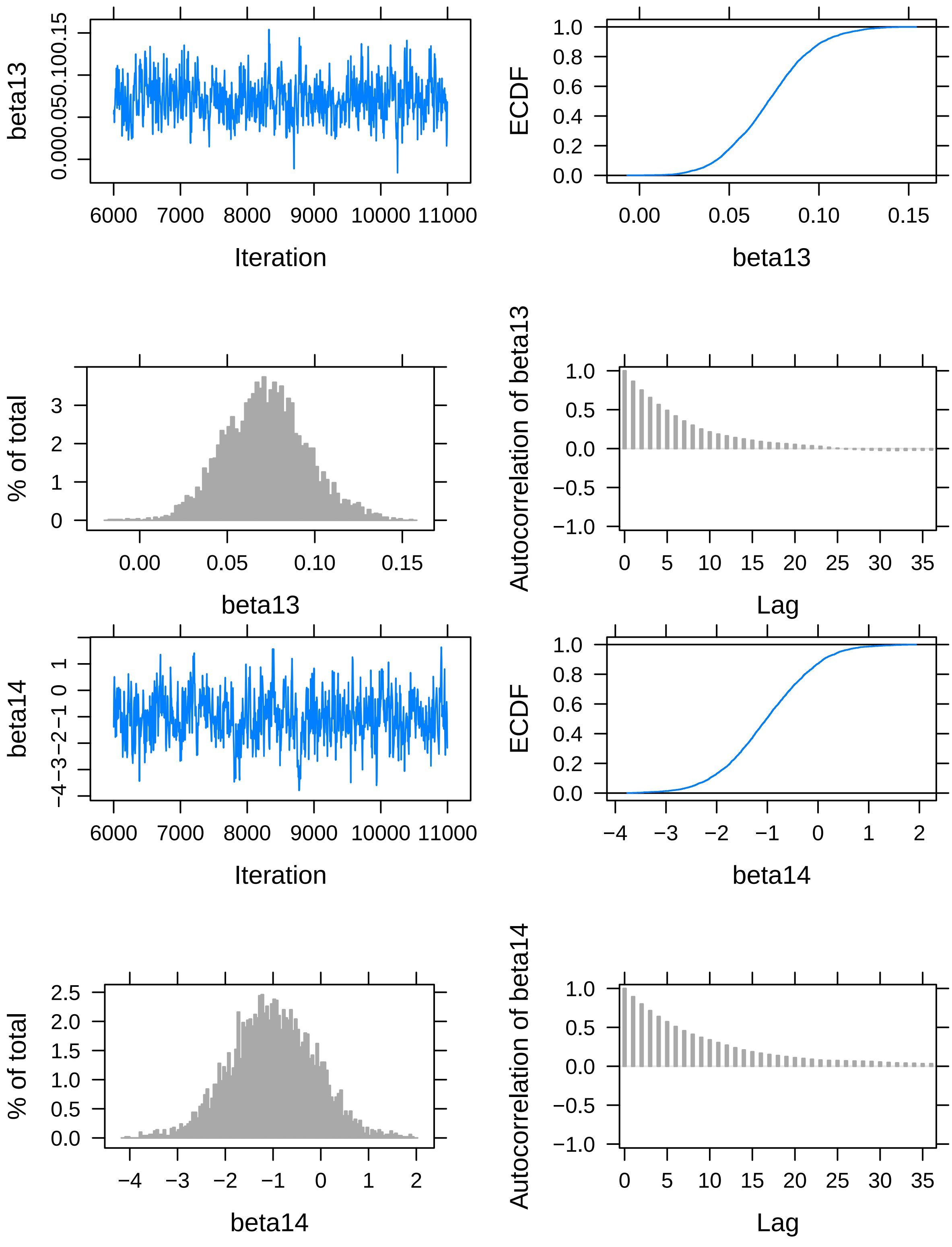
```
## beta21 0.017843870      2.3  1913  0.010869948    NA
```

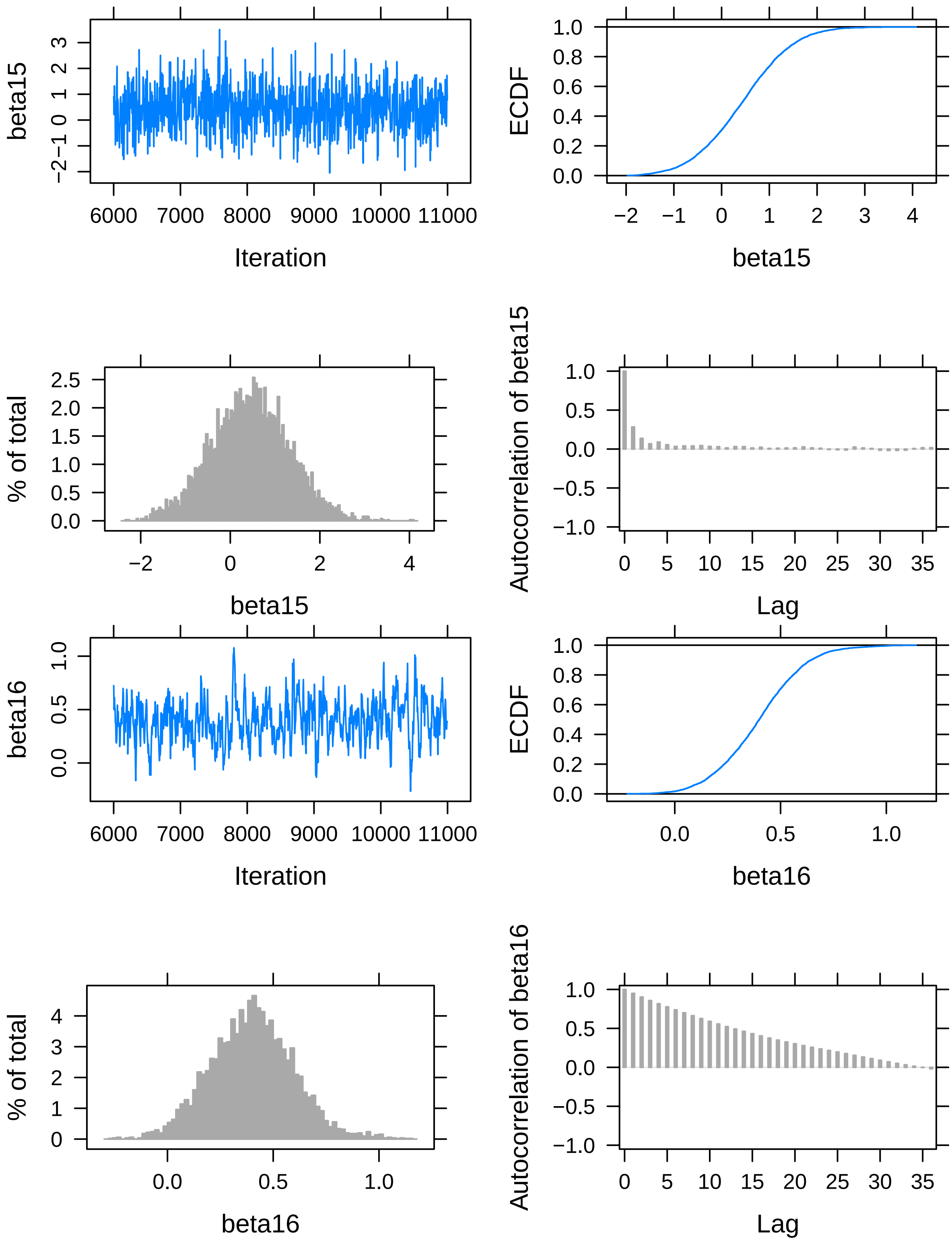
```r
plot(posterior_MLR, vars = "beta1")
```

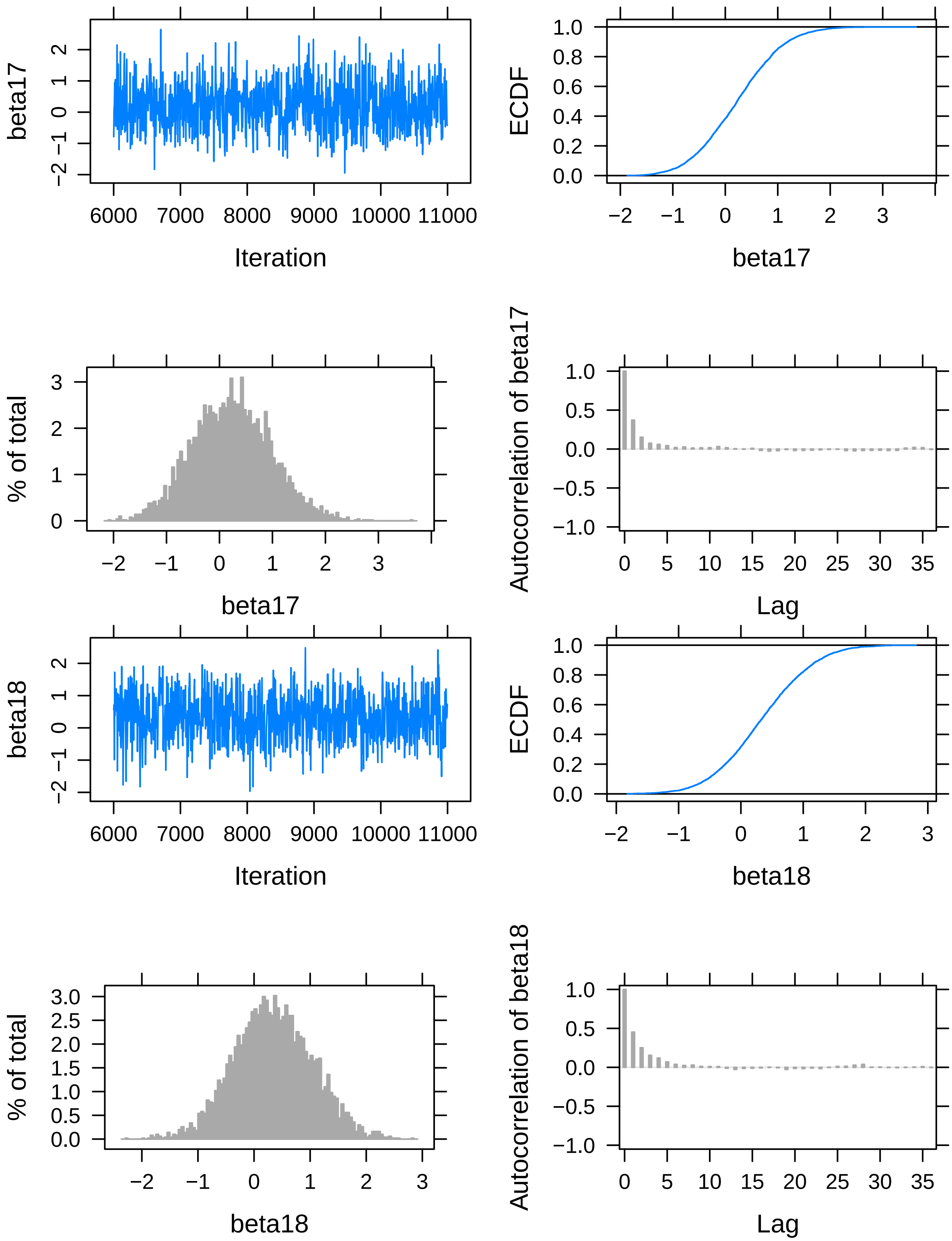
```
## Generating plots...
```

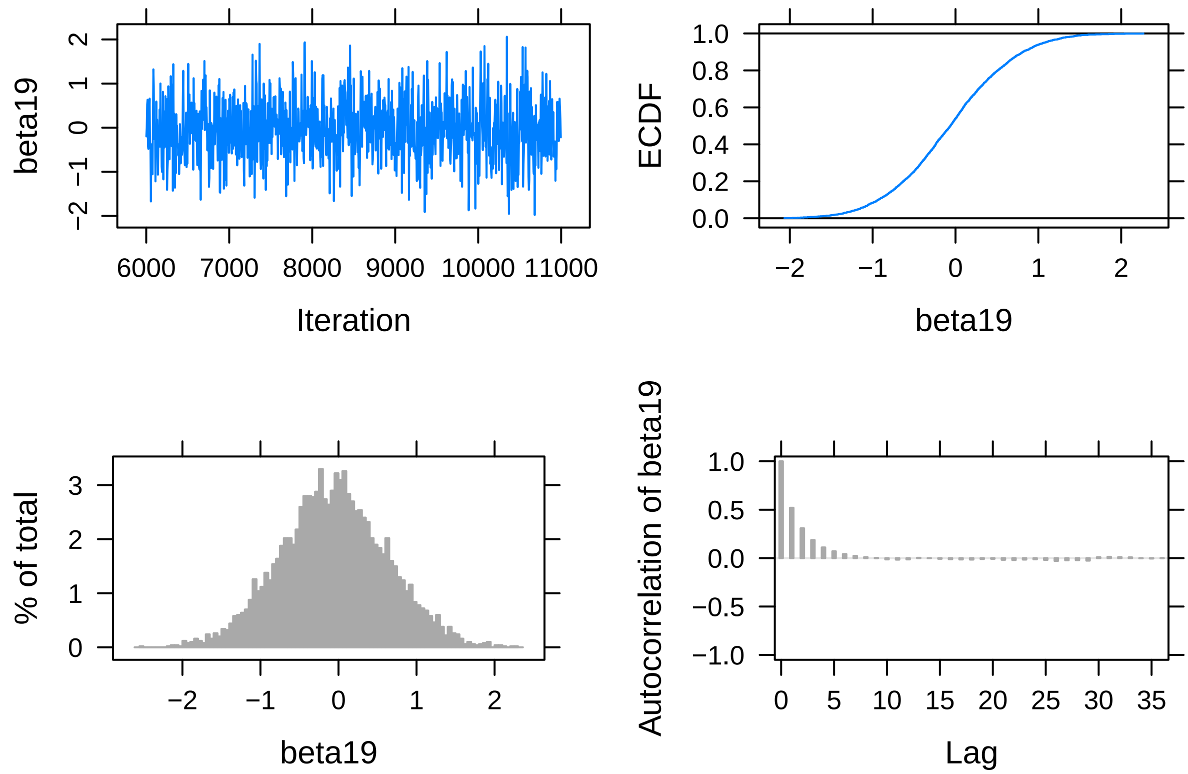


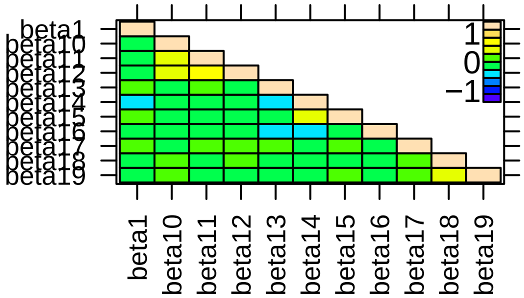

beta11
Iteration
ECDF
beta11
% of total
beta11
Autocorrelation of beta11
Lag
beta12
Iteration
ECDF
beta12
% of total
beta12
Autocorrelation of beta12
Lag


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


beta15
Iteration
6000 7000 8000 9000 10000 11000
−2 −1 0 1 2 3
ECDF
beta15
0.0 0.2 0.4 0.6 0.8 1.0
−2 −1 0 1 2 3 4
% of total
beta15
0.0 0.5 1.0 1.5 2.0 2.5
−2 0 2 4
Autocorrelation of beta15
Lag
−1.0 −0.5 0.0 0.5 1.0
0 5 10 15 20 25 30 35
beta16
Iteration
0.0 0.5 1.0
6000 7000 8000 9000 10000 11000
ECDF
beta16
0.0 0.2 0.4 0.6 0.8 1.0
0.0 0.5 1.0
% of total
beta16
0 1 2 3 4
0.0 0.5 1.0
Autocorrelation of beta16
Lag
−1.0 −0.5 0.0 0.5 1.0
0 5 10 15 20 25 30 35


beta17
Iteration
ECDF
beta17
% of total
beta17
Autocorrelation of beta17
Lag
beta18
Iteration
ECDF
beta18
% of total
beta18
Autocorrelation of beta18
Lag

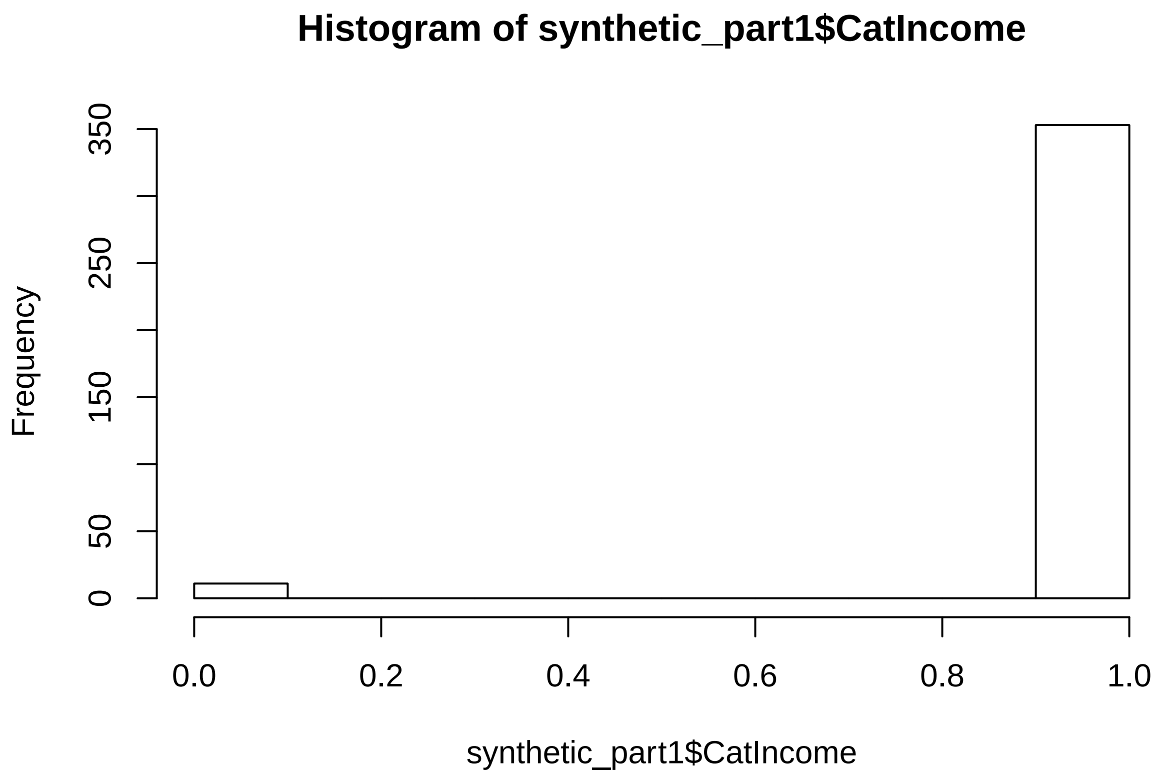




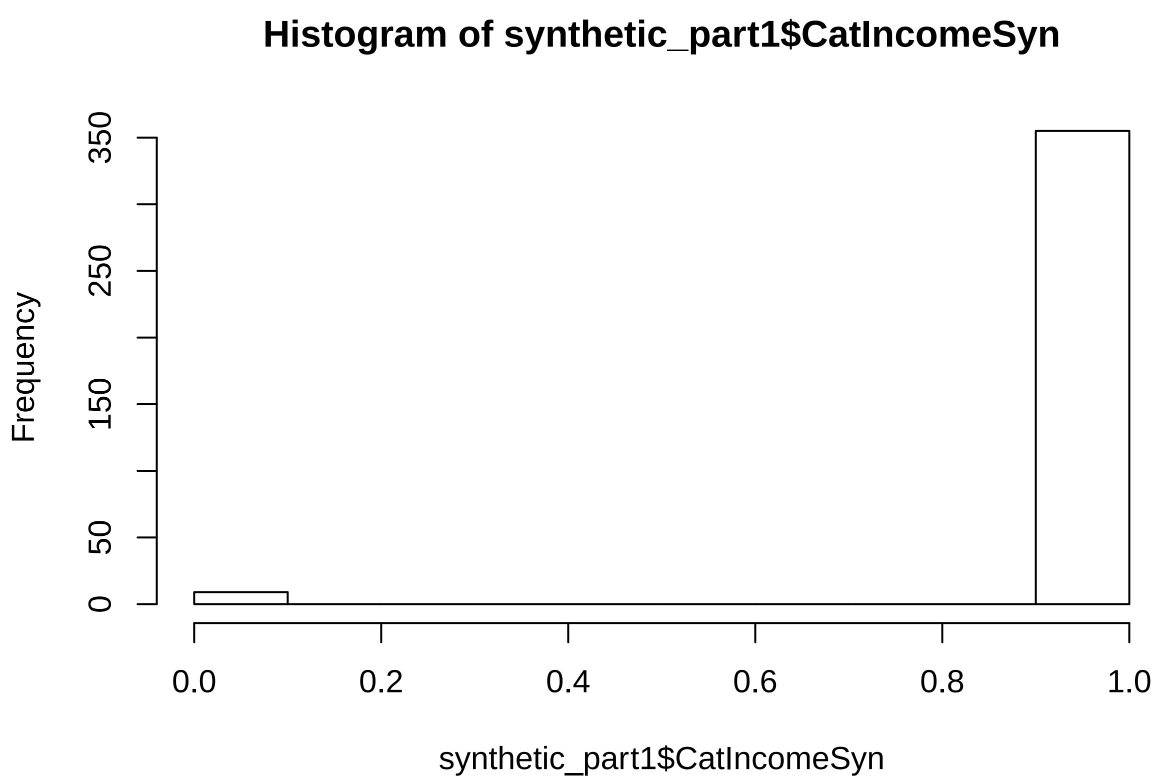

```
## Saving posterior parameter draws
post <- as.mcmc(posterior_MLR)
```

```
## Generating one set of sythetic data
synthesize_catIncome <- 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[in
     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_part1 <- synthesize_catIncome(params, 1, n)
names(synthetic_part1) <- c("CatIncome", "CatIncomeSyn")
```

```
hist(synthetic_part1$CatIncome)
```




```r
hist(synthetic_part1$CatIncomeSyn)
```




```r
## Remove unwanted columns
data$RACEA <- NULL
data$EDUCREC2 <- NULL
```

```r
data$POORYN <- NULL
data$EARNIMP1 <- NULL
data$USUALPL <- NULL
data$DELAYCOST <- NULL
data$HINOTCOVE <- NULL
data$ALCDAYSWK <- NULL
data$CIGDAYMO <- NULL
data$DEPFREQ <- NULL
data$WORFREQ <- NULL
data$INCOME <- NULL
# Expand dataframe columns
data$SEX_1 <- data$SEX$.data_1
data$SEX_2 <- data$SEX$.data_2
data$SEX <- NULL
data$RACE_1 <- data$RACE$.data_1
data$RACE_2 <- data$RACE$.data_2
data$RACE_3 <- data$RACE$.data_3
data$RACE_4 <- data$RACE$.data_4
data$RACE_5 <- data$RACE$.data_5
data$RACE <- NULL
data$EDUC_1 <- data$EDUC$.data_1
data$EDUC_2 <- data$EDUC$.data_2
data$EDUC_3 <- data$EDUC$.data_3
data$EDUC <- NULL
data$HEALTH_1 <- data$HEALTH$.data_1
data$HEALTH_2 <- data$HEALTH$.data_2
data$HEALTH <- NULL
data$WORRY_1 = data$WORRY$.data_1
data$WORRY_2 = data$WORRY$.data_2
data$WORRY_3 = data$WORRY$.data_3
data$WORRY_4 = data$WORRY$.data_4
data$WORRY_5 = data$WORRY$.data_5
data$WORRY <- NULL
## Bind CatIncome to original data
data_org1 <- cbind(data, synthetic_part1$CatIncome)
## Bind CatIncomeSyn to synthetic data
data_syn1 <- cbind(data, synthetic_part1$CatIncomeSyn)
## Rename CatIncome and CatIncomeSyn
colnames(data_org1)[colnames(data_org1) == "synthetic_part1$CatIncome"] <- "INCOME"
colnames(data_syn1)[colnames(data_syn1) == "synthetic_part1$CatIncomeSyn"] <- "INCOME"
colnames(data_org1)
```

```
##  [1] "AGE"          "HOURSWRK"     "EARNIMPOINT1" "HRSLEEP"
##  [5] "SEX_1"        "SEX_2"        "RACE_1"       "RACE_2"
##  [9] "RACE_3"       "RACE_4"       "RACE_5"       "EDUC_1"
## [13] "EDUC_2"       "EDUC_3"       "HEALTH_1"     "HEALTH_2"
## [17] "WORRY_1"      "WORRY_2"      "WORRY_3"      "WORRY_4"
## [21] "WORRY_5"      "INCOME"
```

```r
colnames(data_syn1)
```

```
##  [1] "AGE"          "HOURSWRK"     "EARNIMPOINT1" "HRSLEEP"
##  [5] "SEX_1"        "SEX_2"        "RACE_1"       "RACE_2"
##  [9] "RACE_3"       "RACE_4"       "RACE_5"       "EDUC_1"
```

```
## [13] "EDUC_2"       "EDUC_3"       "HEALTH_1"     "HEALTH_2"
## [17] "WORRY_1"      "WORRY_2"      "WORRY_3"      "WORRY_4"
## [21] "WORRY_5"      "INCOME"
```

Utility evaluation - Global measures

```
n <- dim(data_org1)[1]
merged_data1 <- rbind(data_org1,data_syn1)
merged_data1$S <- c(rep(0,n),rep(1,n))
# Propensity score (note we can't really take the log because of zero income values)
log_reg <- glm(S ~ AGE + HOURSWRK + HRSLEEP + SEX_1 + SEX_2 + RACE_1 + RACE_2 + RACE_3 + RACE_4 + RACE_5
pred <- predict(log_reg, data = merged_data1)
probs <- pred/(1+pred)
Up <- 1/(2*n)*sum((probs - 1/2)^2)
Up
```

```
## [1] 0.2533453
```

```
# Cluster analysis
clusters <- hclust(dist(merged_data1[,1:21]), method = 'average')
G <- 5
clusterCut <-cutree(clusters,G)
cluster_S <- as.data.frame(cbind(clusterCut, merged_data1$S))
names(cluster_S) <- c("cluster","S")
table(cluster_S)
```

```
##        S
## cluster   0   1
##       1 205 205
##       2  72  72
##       3  50  50
##       4  23  23
##       5  14  14
```

```
n_gS <- table(cluster_S)[,1]
n_g <- rowSums(table(cluster_S))
w_g <- n_g / (2*n)
Uc <- (1/G) * sum(w_g * (n_gS/n_g - 1/2)^2)
Uc
```

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

Emperical CDF

```
ecdf_orig <- ecdf(data_org1$INCOME)
ecdf_syn <- ecdf(data_syn1$INCOME)
percentile_orig <- ecdf_orig(merged_data1$INCOME)
percentile_syn <- ecdf_syn(merged_data1$INCOME)
ecdf_diff <- percentile_orig - percentile_syn
Um <- max(abs(ecdf_diff))
Um
```

```
## [1] 0.005494505
```

```r
Ua <- mean(ecdf_diff^2)
Ua
```

```
## [1] 8.293844e-07
```

1) Take all rows where income was syn to 1 above
2) Take all rows where income was orig non-zero
3) Log 2), use it in JAGS
4) Use model from 3) to syn all rows in 1)
5) Combine and evaluate data

Part 2: Synthetic Linear Regression Model (syn income given all non-zero entries from part 1)

```r
## JAGS script
modelString_part2 <-"
model {
## sampling
for (i in 1:N){
y[i] ~ dnorm(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], 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)
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))
}"
```

```
dataNoZeros = data[!data$EARNIMPOINT1 == 0,]
y = log(dataNoZeros$EARNIMPOINT1)
x_age = as.vector(dataNoZeros$AGE) ## age
x_sex_male = as.vector(dataNoZeros$SEX_1) ## male
x_sex_female = as.vector(dataNoZeros$SEX_2) ## female
x_race_w = as.vector(dataNoZeros$RACE_1) ## white
x_race_b = as.vector(dataNoZeros$RACE_2) ## black/african-american
x_race_i = as.vector(dataNoZeros$RACE_3) ## american indian
x_race_a = as.vector(dataNoZeros$RACE_4) ## asian
x_race_o = as.vector(dataNoZeros$RACE_5) ## other races
x_educ_1 = as.vector(dataNoZeros$EDUC_3) ## 4 years of high school or less
x_educ_2 = as.vector(dataNoZeros$EDUC_1) ## 4 years of college
x_educ_3 = as.vector(dataNoZeros$EDUC_2) ## 5+ years of college
x_hourswrk = as.vector(dataNoZeros$HOURSWRK) ## hours of work
x_health_cov = as.vector(dataNoZeros$HEALTH_1) ## has health coverage
x_health_nocov = as.vector(dataNoZeros$HEALTH_2) ## has no health coverage
x_hrsleep = as.vector(dataNoZeros$HRSLEEP) ## hours of sleep
x_wor_daily = as.vector(dataNoZeros$WORRY_2) ## worry daily
x_wor_weekly = as.vector(dataNoZeros$WORRY_5) ## worry weekly
x_wor_monthly = as.vector(dataNoZeros$WORRY_4) ## worry monthly
x_wor_fewtimes = as.vector(dataNoZeros$WORRY_3) ## worry few times a year
x_wor_never = as.vector(dataNoZeros$WORRY_1) ## worry never
N = length(y) # Compute the number of observations
## Pass the data and hyperparameter values to JAGS
the_data_part2 <- 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,
"a" = 1, "b" = 1)
```

```r
## Run the JAGS code for this model:
posterior_MLR <- run.jags(modelString_part2,
n.chains = 1,
data = the_data_part2,
monitor = c("beta0", "beta1", "beta2",
"beta3", "beta4", "beta5",
"beta6", "beta7", "beta8", "beta10",
"beta11", "beta12", "beta13", "beta14", "beta15", "beta16", "beta17",
"beta18", "beta19", "beta20", "beta21", "sigma"),
adapt = 1000,
burnin = 5000,
sample = 5000,
thin = 20,
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 100000 iterations...
## Simulation complete
## Calculating summary statistics...

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

## Finished running the simulation
```

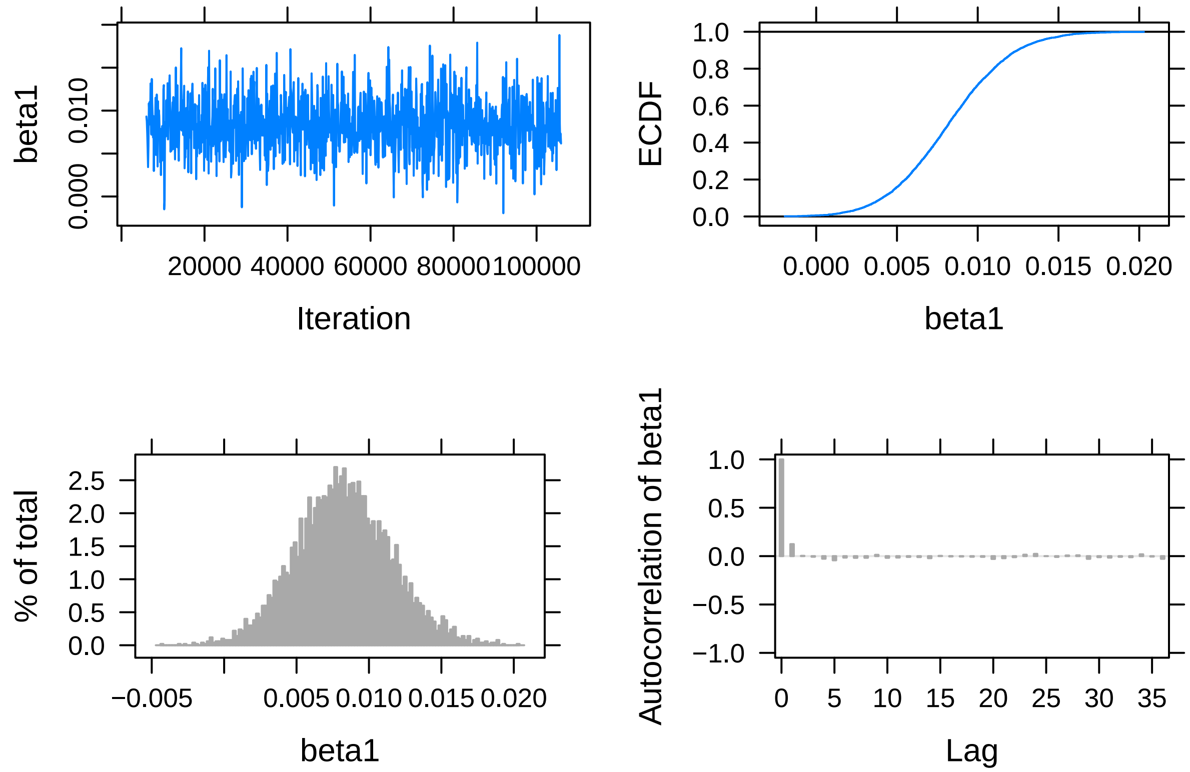
```r
## JAGS output
summary(posterior_MLR)
```

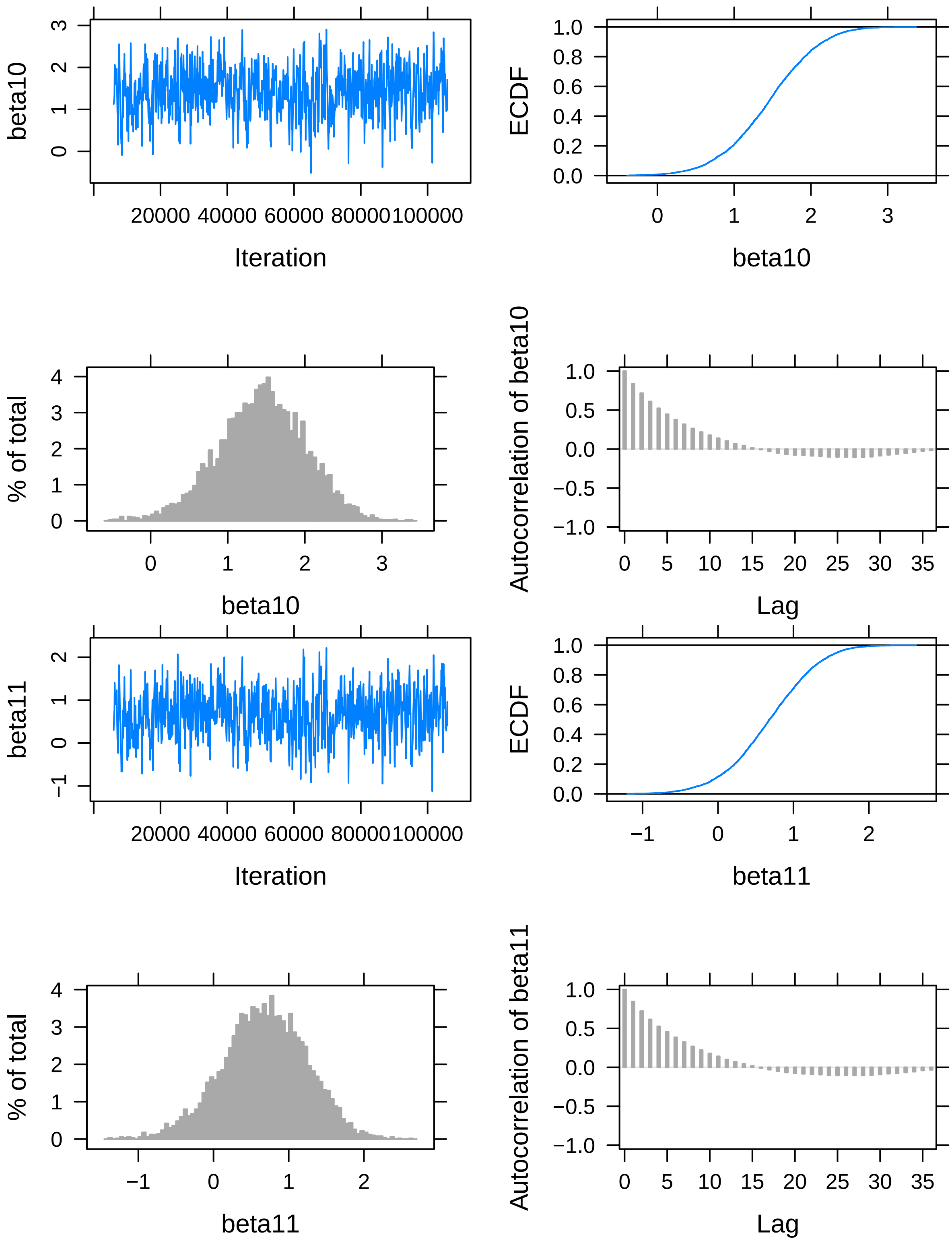
```
##             Lower95       Median    Upper95          Mean          SD Mode
## beta0   1.430506237  3.085194245 4.61676840  3.0850861639 0.831035218   NA
## beta1   0.002279226  0.008190387 0.01537494  0.0082444217 0.003298690   NA
## beta2   0.451834626  1.656501040 3.01685633  1.6761524873 0.665269216   NA
## beta3   0.251833570  1.435298750 2.83833950  1.4593425229 0.665415542   NA
## beta4  -0.257988363  0.678961472 1.58555554  0.6827904107 0.461493781   NA
## beta5  -0.514091328  0.373489892 1.38805097  0.3719123851 0.474491880   NA
## beta6  -0.698051996  0.348994558 1.26949354  0.3357819863 0.507741431   NA
## beta7  -0.161091390  0.797161427 1.78072116  0.7987097203 0.490995286   NA
## beta8  -0.353266136  0.868099163 2.11555442  0.8698396570 0.627772764   NA
## beta10  0.333657574  1.452590642 2.54219691  1.4396551401 0.564917732   NA
## beta11 -0.468480728  0.679049854 1.70938301  0.6754678139 0.561739772   NA
## beta12 -0.062604101  1.034443534 2.11003620  1.0267241393 0.560862388   NA
## beta13  0.024603972  0.031515798 0.03833936  0.0314901687 0.003495631   NA
## beta14  0.490702345  1.903232444 3.10675955  1.8990919277 0.646837008   NA
## beta15  0.109789537  1.414634657 2.70075713  1.4167627518 0.645332045   NA
## beta16 -0.070138464 -0.001569241 0.06853656 -0.0009068957 0.035862043   NA
## beta17 -0.221141571  0.663445703 1.54556554  0.6663222228 0.453336642   NA
## beta18 -0.218430056  0.676489603 1.53513665  0.6846231118 0.449306163   NA
## beta19 -0.195410402  0.679806755 1.56434693  0.6870664682 0.449529211   NA
## beta20 -0.242308283  0.645493942 1.55337876  0.6594888591 0.460839555   NA
## beta21 -0.367862969  0.486496845 1.42155829  0.4911985237 0.461302371   NA
## sigma   0.815909344  0.884814081 0.95098029  0.8862995979 0.034149594   NA
##               MCerr MC%ofSD SSeff        AC.200 psrf
## beta0  5.834034e-02     7.0   203  0.4493562174   NA
## beta1  4.875218e-05     1.5  4578 -0.0177447141   NA
```

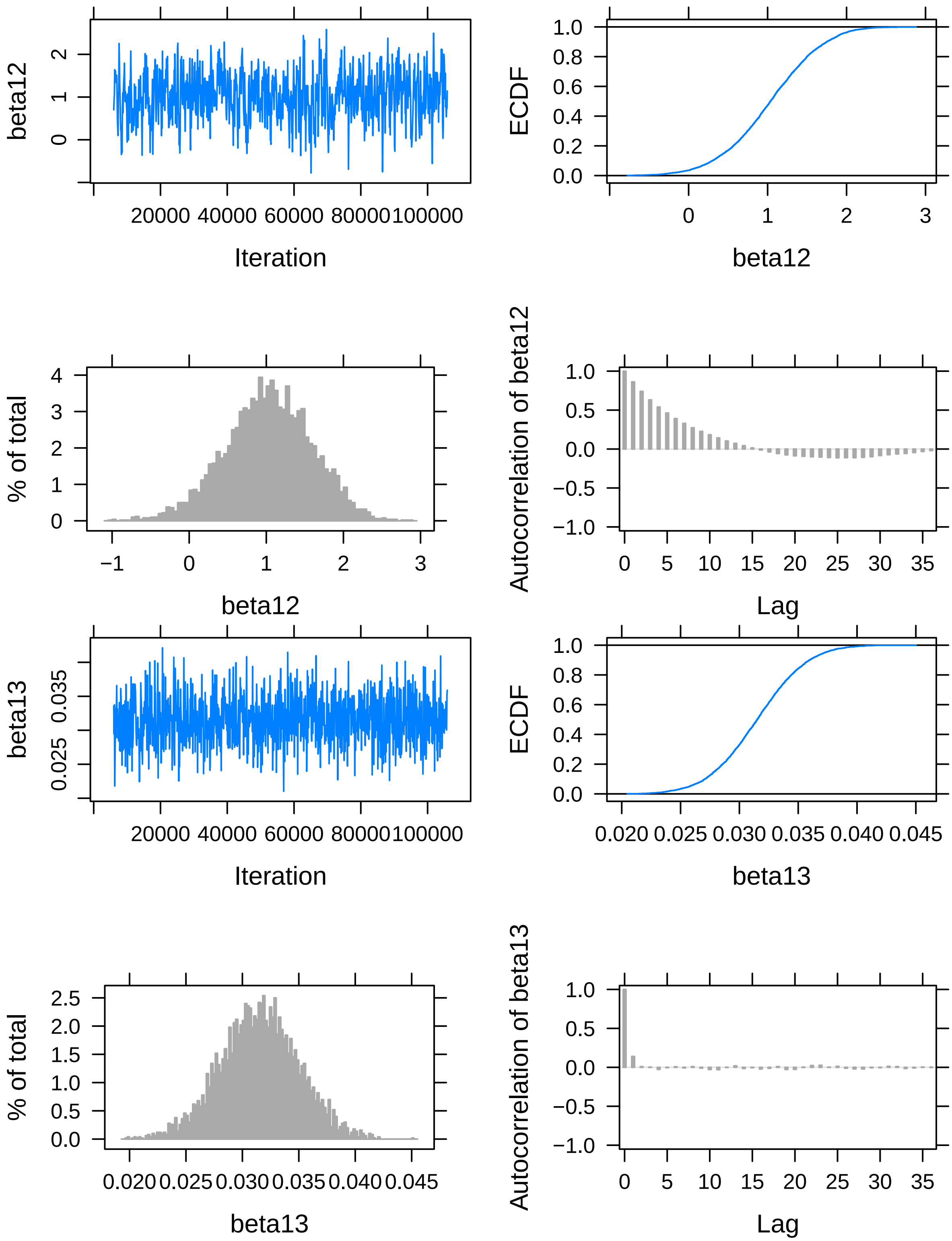
```
## beta2  4.097256e-02      6.2    264  0.2955147317    NA
## beta3  4.133762e-02      6.2    259  0.2978056007    NA
## beta4  1.871488e-02      4.1    608  0.0304440058    NA
## beta5  1.928702e-02      4.1    605  0.0310264536    NA
## beta6  1.809277e-02      3.6    788  0.0092630427    NA
## beta7  1.392278e-02      2.8   1244  0.0214926893    NA
## beta8  1.426237e-02      2.3   1937  0.0048687699    NA
## beta10 2.848088e-02      5.0    393  0.1774744940    NA
## beta11 2.409544e-02      4.3    544  0.1790081639    NA
## beta12 2.918045e-02      5.2    369  0.1832801911    NA
## beta13 5.692429e-05      1.6   3771 -0.0286529975    NA
## beta14 3.804836e-02      5.9    289  0.2685461917    NA
## beta15 3.324490e-02      5.2    377  0.2564965515    NA
## beta16 9.080225e-04      2.5   1560 -0.0266764299    NA
## beta17 1.780931e-02      3.9    648  0.0381573200    NA
## beta18 1.787244e-02      4.0    632  0.0360038474    NA
## beta19 1.786828e-02      4.0    633  0.0326900332    NA
## beta20 1.770012e-02      3.8    678  0.0314632042    NA
## beta21 1.782909e-02      3.9    669  0.0427400087    NA
## sigma  4.829482e-04      1.4   5000 -0.0002367632    NA
```

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

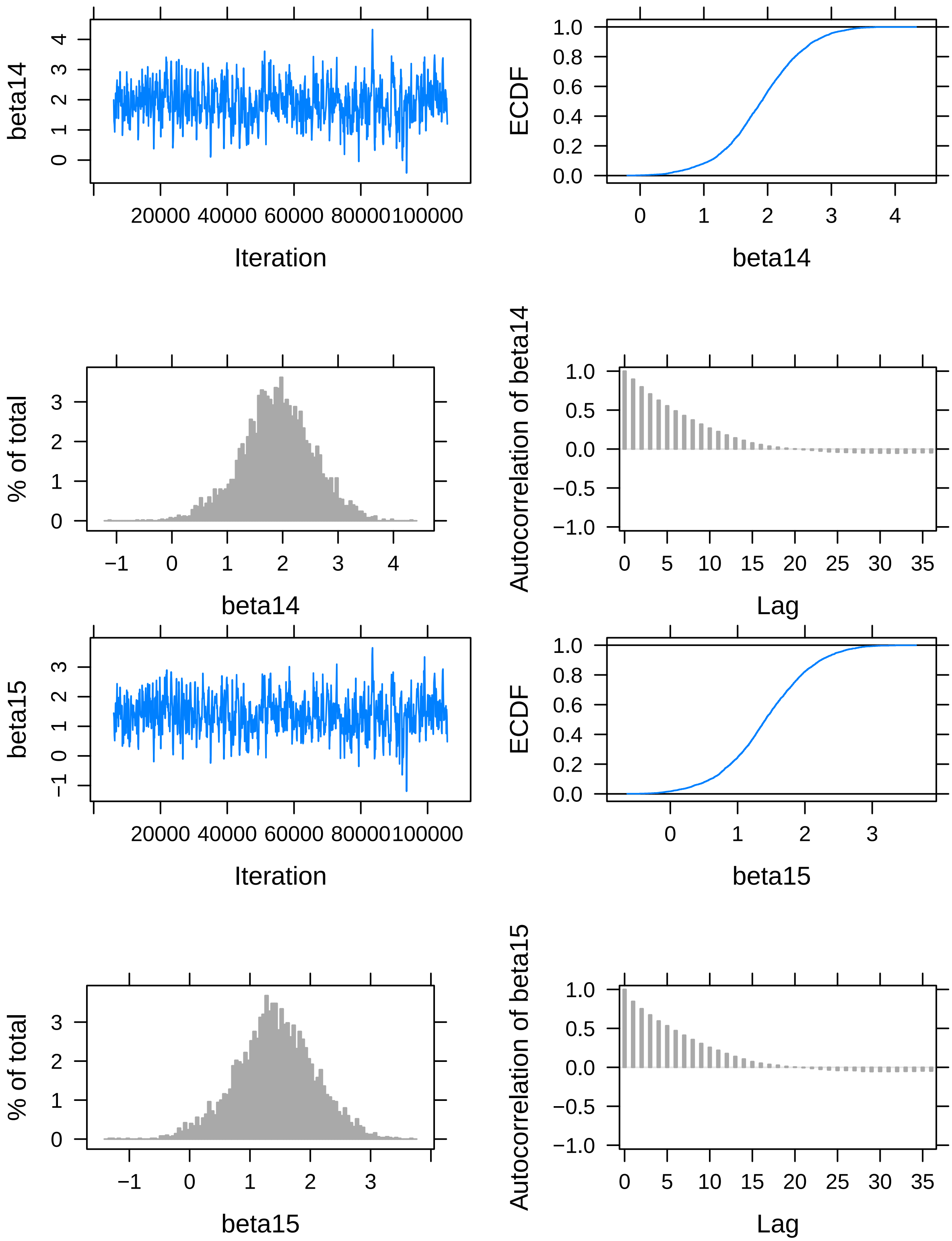
```
## Generating plots...
```

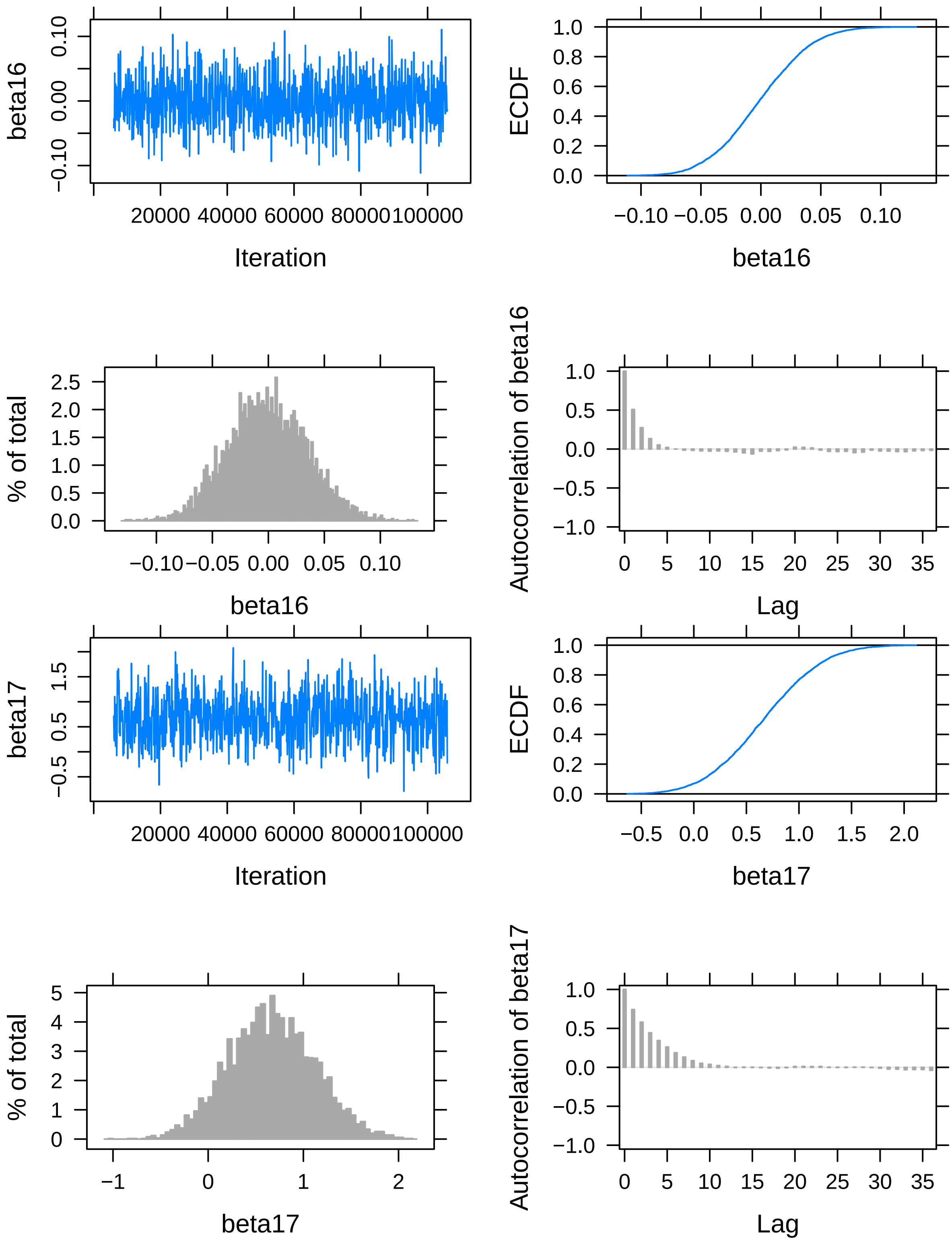


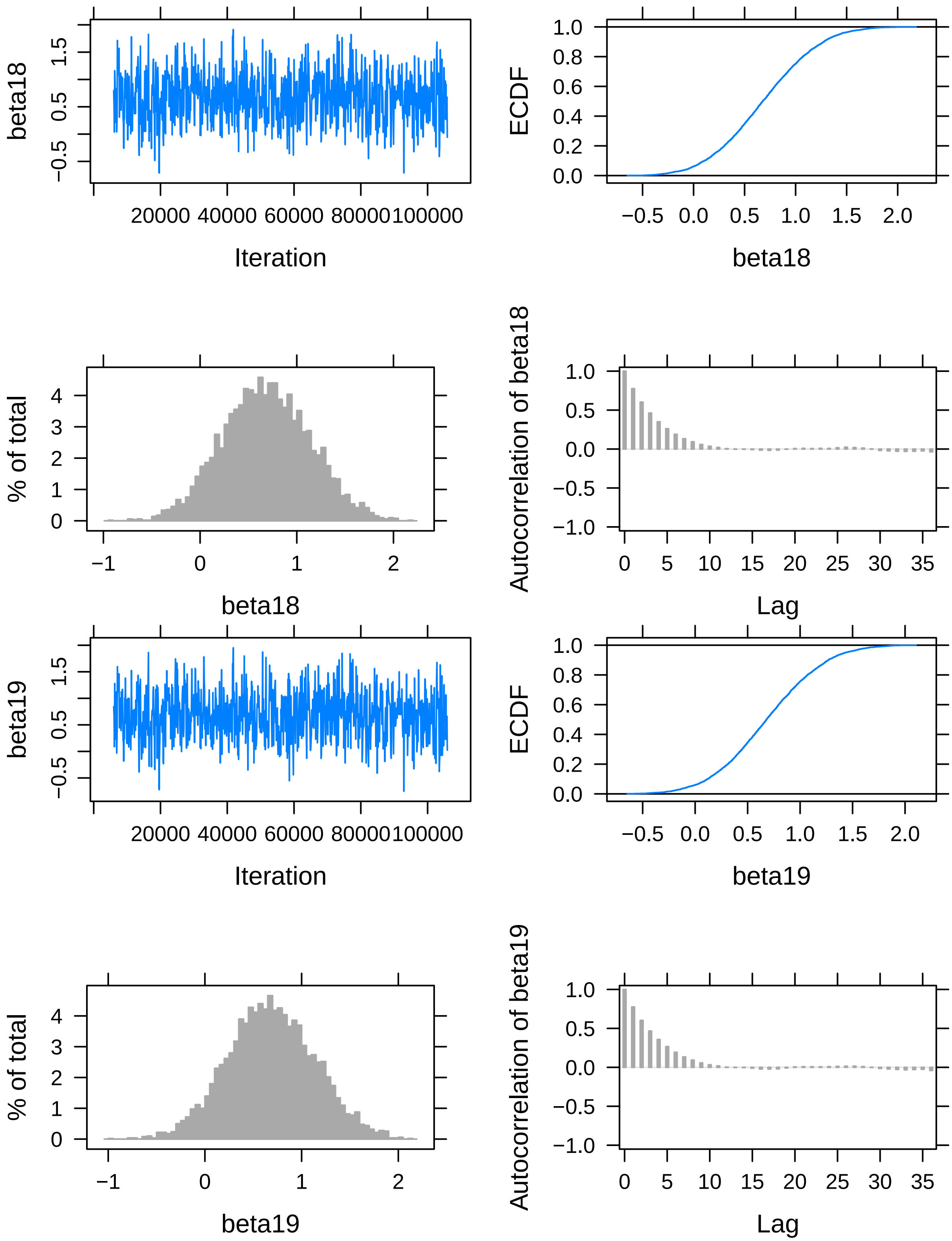

beta10
Iteration
20000 40000 60000 80000100000
ECDF
beta10
% of total
beta10
Autocorrelation of beta10
Lag
beta11
Iteration
20000 40000 60000 80000100000
ECDF
beta11
% of total
beta11
Autocorrelation of beta11
Lag

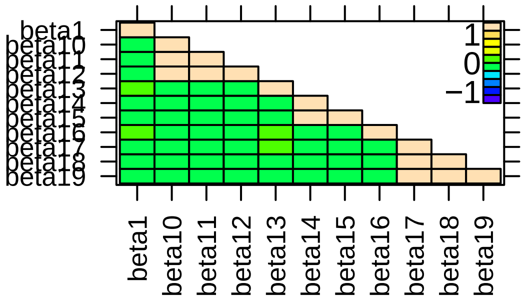


beta12
Iteration
20000 40000 60000 80000 100000
ECDF
beta12
% of total
beta12
Autocorrelation of beta12
Lag
beta13
Iteration
ECDF
beta13
% of total
beta13
Autocorrelation of beta13
Lag


beta14
Iteration
ECDF
beta14
% of total
beta14
Autocorrelation of beta14
Lag
beta15
Iteration
ECDF
beta15
% of total
beta15
Autocorrelation of beta15
Lag


beta16
0.10
0.00
−0.10
20000 40000 60000 80000100000
Iteration
ECDF
1.0
0.8
0.6
0.4
0.2
0.0
−0.10 −0.05 0.00 0.05 0.10
beta16
% of total
2.5
2.0
1.5
1.0
0.5
0.0
−0.10−0.05 0.00 0.05 0.10
beta16
Autocorrelation of beta16
1.0
0.5
0.0
−0.5
−1.0
0 5 10 15 20 25 30 35
Lag
beta17
1.5
0.5
−0.5
20000 40000 60000 80000100000
Iteration
ECDF
1.0
0.8
0.6
0.4
0.2
0.0
−0.5 0.0 0.5 1.0 1.5 2.0
beta17
% of total
5
4
3
2
1
0
−1 0 1 2
beta17
Autocorrelation of beta17
1.0
0.5
0.0
−0.5
−1.0
0 5 10 15 20 25 30 35
Lag


beta18
Iteration
20000 40000 60000 80000100000
−0.5 0.5 1.5
ECDF
0.0 0.2 0.4 0.6 0.8 1.0
−0.5 0.0 0.5 1.0 1.5 2.0
beta18
% of total
0 1 2 3 4
−1 0 1 2
beta18
Autocorrelation of beta18
−1.0 −0.5 0.0 0.5 1.0
0 5 10 15 20 25 30 35
Lag
beta19
Iteration
20000 40000 60000 80000100000
−0.5 0.5 1.5
ECDF
0.0 0.2 0.4 0.6 0.8 1.0
−0.5 0.0 0.5 1.0 1.5 2.0
beta19
% of total
0 1 2 3 4
−1 0 1 2
beta19
Autocorrelation of beta19
−1.0 −0.5 0.0 0.5 1.0
0 5 10 15 20 25 30 35
Lag



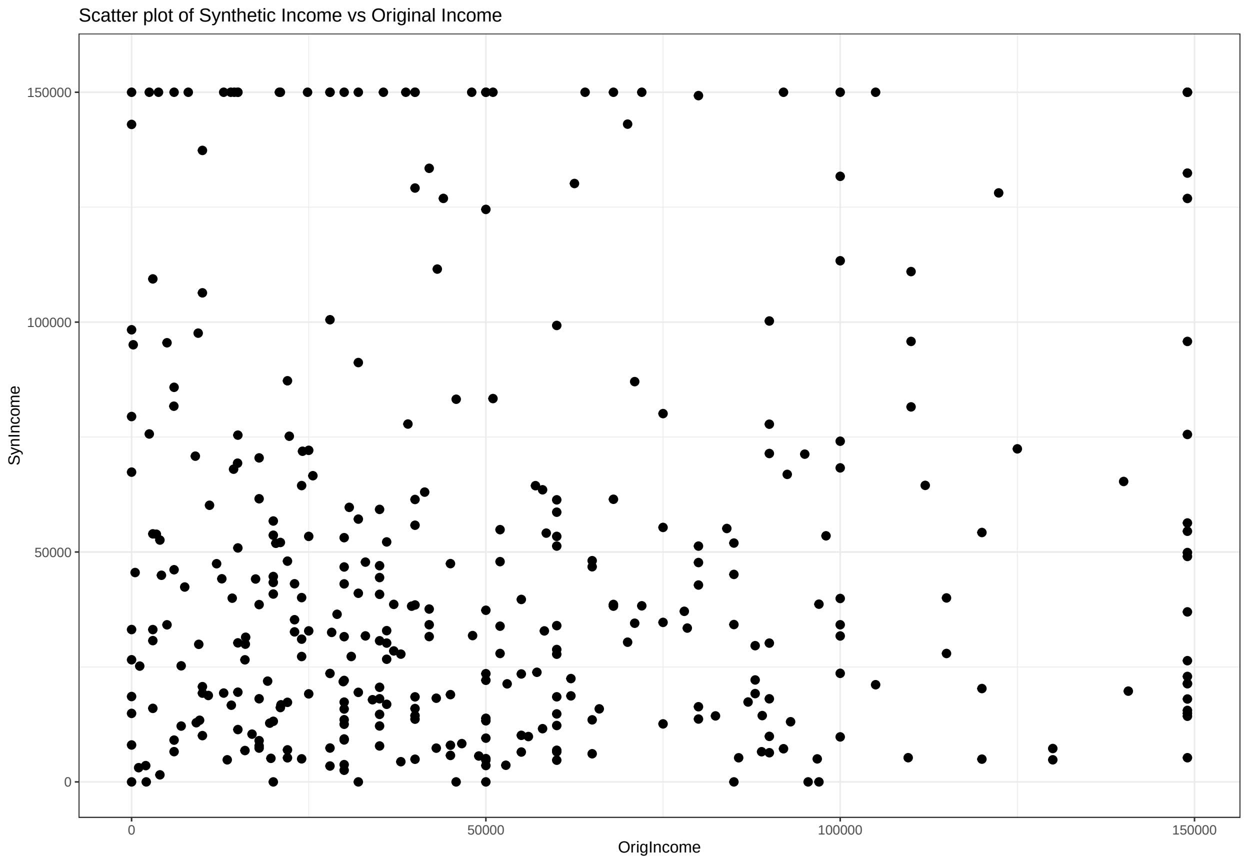

```
## Saving posterior parameter draws
post <- as.mcmc(posterior_MLR)
```

```
## Generating one set of sythetic data
synthesize <- function(X, index, n){
  mean_Y <- post[index, "beta0"] +  X$x_age * post[index, "beta1"] +  X$x_sex_male * post[index, "beta2
  synthetic_Y <- rnorm(n, mean_Y, post[index, "sigma"])
  data.frame(data$EARNIMPOINT1, exp(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("OrigIncome", "SynIncome")
```

```
for (i in 1:nrow(synthetic_one)){
  if (synthetic_part1$CatIncomeSyn[i] == 0) {
    synthetic_one$SynIncome[i] = 0
  }
  if (synthetic_one$SynIncome[i] > 150000){
    synthetic_one$SynIncome[i] = 150000
  }
}
```

```
ggplot(synthetic_one, aes(x = OrigIncome, y = SynIncome)) +
  geom_point(size = 1) +
  labs(title = "Scatter plot of Synthetic Income vs Original Income") +
  theme_bw(base_size = 6, base_family = "") +
  coord_cartesian(xlim = c(0, 155000)) +
  coord_cartesian(ylim = c(0, 155000))
```

```
## Coordinate system already present. Adding new coordinate system, which will replace the existing one
```




```r
## Bind CatIncome to original data
data_org <- cbind(data, synthetic_one$OrigIncome)
## Bind CatIncomeSyn to synthetic data
data_syn <- cbind(data, synthetic_one$SynIncome)
## Rename CatIncome and CatIncomeSyn
colnames(data_org)[colnames(data_org) == "synthetic_one$OrigIncome"] <- "INCOME"
colnames(data_syn)[colnames(data_syn) == "synthetic_one$SynIncome"] <- "INCOME"
colnames(data_org)
```

```
##  [1] "AGE"          "HOURSWRK"     "EARNIMPOINT1" "HRSLEEP"
##  [5] "SEX_1"        "SEX_2"        "RACE_1"       "RACE_2"
##  [9] "RACE_3"       "RACE_4"       "RACE_5"       "EDUC_1"
## [13] "EDUC_2"       "EDUC_3"       "HEALTH_1"     "HEALTH_2"
## [17] "WORRY_1"      "WORRY_2"      "WORRY_3"      "WORRY_4"
## [21] "WORRY_5"      "INCOME"
```

```r
colnames(data_syn)
```

```
##  [1] "AGE"          "HOURSWRK"     "EARNIMPOINT1" "HRSLEEP"
##  [5] "SEX_1"        "SEX_2"        "RACE_1"       "RACE_2"
##  [9] "RACE_3"       "RACE_4"       "RACE_5"       "EDUC_1"
## [13] "EDUC_2"       "EDUC_3"       "HEALTH_1"     "HEALTH_2"
## [17] "WORRY_1"      "WORRY_2"      "WORRY_3"      "WORRY_4"
## [21] "WORRY_5"      "INCOME"
```

TODO: Utility evaluation -global measures 1) propensity score (5) 2) cluster analysis (5) 3) emperical cdf (5) -analysis specific measures 4) mean, median, etc. (6) 5) syn data variability (multiple syn datasets generated) (6) 6) interval overlap (6) Risk evaluation -identification disclosure 7) expected match risk (7) 8) true match rate (7) 9) false match rate (7) -attribute risk disclosure 10) AR disclosure CatIncome (do ?): Currently working: Need to implement: Final model (do all): Currently working: 1,2,3,4,5,6,7,8,9 Need to implement: 10 # Utility evaluation - Global measures

```
n <- dim(data_org)[1]
merged_data <- rbind(data_org,data_syn)
merged_data$S <- c(rep(0,n),rep(1,n))
# Propensity score (note we can't really take the log because of zero income values)
log_reg <- glm(S ~ AGE + HOURSWRK + HRSLEEP + SEX_1 + SEX_2 + RACE_1 + RACE_2 + RACE_3 + RACE_4 + RACE_5
pred <- predict(log_reg, data = merged_data)
probs <- pred/(1+pred)
Up <- 1/(2*n)*sum((probs - 1/2)^2)
Up
```

```
## [1] 0.2527945
```

```
# Cluster analysis
clusters <- hclust(dist(merged_data[,1:21]), method = 'average')
G <- 5
clusterCut <-cutree(clusters,G)
cluster_S <- as.data.frame(cbind(clusterCut, merged_data$S))
names(cluster_S) <- c("cluster","S")
table(cluster_S)
```

```
##        S
## cluster   0   1
##       1 205 205
##       2  72  72
##       3  50  50
##       4  23  23
##       5  14  14
```

```
n_gS <- table(cluster_S)[,1]
n_g <- rowSums(table(cluster_S))
w_g <- n_g / (2*n)
Uc <- (1/G) * sum(w_g * (n_gS/n_g - 1/2)^2)
Uc
```

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

Emperical CDF

```
ecdf_orig <- ecdf(data_org$INCOME)
ecdf_syn <- ecdf(data_syn$INCOME)
percentile_orig <- ecdf_orig(merged_data$INCOME)
percentile_syn <- ecdf_syn(merged_data$INCOME)
ecdf_diff <- percentile_orig - percentile_syn
Um <- max(abs(ecdf_diff))
Um
```

```
## [1] 0.09615385
```

```
Ua <- mean(ecdf_diff^2)
Ua
```

```
## [1] 0.002560797
```

Utility evaluation - Analysis specific measures: Mean and median

```
mean(data_org$INCOME)
```

```
## [1] 50536.74
```

```
mean(data_syn$INCOME)
```

```
## [1] 49059.94
```

```
median(data_org$INCOME)
```

```
## [1] 40000
```

```
median(data_syn$INCOME)
```

```
## [1] 34170.72
```

Utility evaluation - Analysis specific measures: Synthetic data variability

```
synthesizeMany <- function(X, index, n){
  mean_Y <- post[index, "beta0"] +  X$x_age * post[index, "beta1"] +  X$x_sex_male * post[index, "beta2
  synthetic_Y <- rnorm(n, mean_Y, post[index, "sigma"])
  data.frame(exp(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_
m <- 20
synthetic_m <- vector("list",m)
for(i in 1:m){
  syn <- synthesizeMany(params,i,n)
  names(syn) <- c("SynIncome")
  for (k in 1:nrow(syn)){
    if (synthetic_part1$CatIncomeSyn[i] == 0) {
      syn$SynIncome[i] = 0
    }
    if (syn$SynIncome[i] > 150000){
      syn$SynIncome[i] = 150000
    }
  }
  synthetic_m[[i]] <- syn
}
q <- rep(NA,m)
v <- rep(NA,m)
for(i in 1:m){
  syn <- synthetic_m[[i]]
  q[i] <- mean(syn$SynIncome)
  v[i] <- var(syn$SynIncome)/n
}
q_bar_m <- mean(q)
b_m <- var(q)
v_bar_m <- mean(v)
T_p <- b_m / m + v_bar_m
```

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

```
## [1] 62457.57
```

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

```
## [1] 52888.25 72026.89
```

```r
mean_org <- mean(data_org$INCOME)
sd_org <- sd(data_org$INCOME)
t_score_org <- qt(p = 0.975, df = n-1)
mean_org
```

```
## [1] 50536.74
```

```r
interval_orig <- c(mean_org - t_score_org * sd_org / sqrt(n),
  mean_org + t_score_org * sd_org / sqrt(n))
interval_orig
```

```
## [1] 46418.68 54654.81
```

Utility evaluation - Analysis specific measures: Interval overlap

```r
L_s <- interval_syn[1]
U_s <- interval_syn[2]
L_o <- interval_orig[1]
U_o <- interval_orig[2]
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
```

```
## [1] 0.1533961
```

Risk evaluation - Identification disclosure: Expected match risk, True match rate, False match rate

```r
CalculateKeyQuantities <- function(origdata, syndata, known.vars, syn.vars, n){
  origdata <- origdata
  syndata <- syndata
  n <- n
  c_vector <- rep(NA, n)
  T_vector <- rep(NA, n)
  for (i in 1:n){
    match <- (eval(parse(text=paste("origdata$",syn.vars,"[i]==
                                     syndata$",syn.vars,sep="",collapse="&")))&
                eval(parse(text=paste("origdata$",known.vars,"[i]==
                                       syndata$",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)
}


 known.vars <- c("SEX_2", "SEX_1", "RACE_2","RACE_1","RACE_5","RACE_4","RACE_3",  "AGE", "EDUC_3", "EDU
 syn.vars <- c("INCOME")
 n <- dim(data_org)[1]
 KeyQuantities <- CalculateKeyQuantities(data_org, data_syn,
                                        known.vars, syn.vars, n)


 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)
}

#- 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)
```

```
ThreeSummaries[["exp_match_risk"]]
```

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

```r
ThreeSummaries[["true_match_rate"]]
```

```
## [1] 0.002747253
```

```r
ThreeSummaries[["false_match_rate"]]
```

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

Risk evaluation - Attribute risk disclosure

```r
# Assume all variables except for income are known
# But we cannot log the values!
```