Methods for Utility Evaluation #2

MATH 301 Data Confidentiality

Henrik Olsson

February 25, 2020

```r
origdata<- read.csv("CEdata.csv")
head(origdata)
```

```
##   UrbanRural Income Race Expenditure
## 1          1  98600    1    5972.167
## 2          1  24360    1    5854.500
## 3          1  80200    1    5506.667
## 4          1 150500    1    8968.891
## 5          1 130000    1   10092.833
## 6          1  32836    1    5520.267
```

Read Drechsler (2001) Chapter 6-1, 7-1 in the References folder, and prepare the following results.

```r
origdata$LogExp <- log(origdata$Expenditure)
origdata$LogIncome <- log(origdata$Income)

## create indicator variable for Rural (2)
origdata$Rural = fastDummies::dummy_cols(origdata$UrbanRural)[,names(fastDummies::dummy_cols(origdata$U
== ".data_1"]

## create indicator variables for Black (3), Native American (4),
## Asian (5), Pacific Islander (6), and Multi-race (7)
origdata$Race_Black = fastDummies::dummy_cols(origdata$Race)[,names(fastDummies::dummy_cols(origdata$Ra
origdata$Race_NA = fastDummies::dummy_cols(origdata$Race)[,names(fastDummies::dummy_cols(origdata$Race)
origdata$Race_Asian = fastDummies::dummy_cols(origdata$Race)[,names(fastDummies::dummy_cols(origdata$Ra
origdata$Race_PI = fastDummies::dummy_cols(origdata$Race)[,names(fastDummies::dummy_cols(origdata$Race)
origdata$Race_M = fastDummies::dummy_cols(origdata$Race)[,names(fastDummies::dummy_cols(origdata$Race))
```

```r
## JAGS script
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))
}"
```

```r
y = as.vector(origdata$LogExp)
x_income = as.vector(origdata$LogIncome)
x_rural = as.vector(origdata$Rural)
x_race_B = as.vector(origdata$Race_Black)
x_race_N = as.vector(origdata$Race_NA)
x_race_A = as.vector(origdata$Race_Asian)
x_race_P = as.vector(origdata$Race_PI)
x_race_M = as.vector(origdata$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)
```

```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", "sigma"),
adapt = 1000,
burnin = 5000,
sample = 5000,
thin = 20,
inits = initsfunction)
```

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

```
## Compiling rjags model...
## Calling the simulation using the rjags method...
## Note: the model did not require adaptation
```

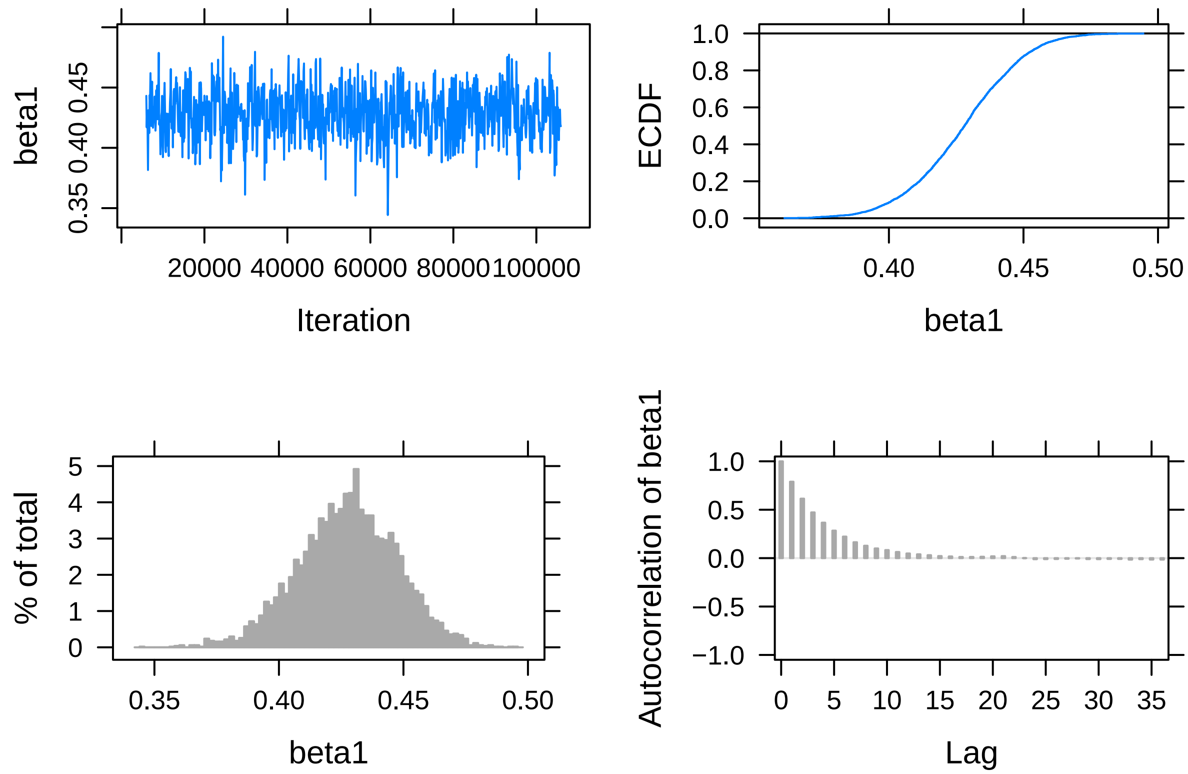
```
## Burning in the model for 5000 iterations...
## Running the model for 100000 iterations...
## Simulation complete
## Calculating summary statistics...

## 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  3.58046793  4.00567714  4.46873420  4.01026369 0.22739281   NA
## beta1  0.38869072  0.42806382  0.46552491  0.42748795 0.01979320   NA
## beta2  0.06829052  0.26991798  0.47936432  0.27011927 0.10454790   NA
## beta3 -0.33605648 -0.19443785 -0.04513397 -0.19471621 0.07412360   NA
## beta4 -0.51146051  0.01343539  0.52291297  0.01371237 0.26926773   NA
## beta5 -0.06626353  0.15835010  0.39460914  0.15949581 0.11802871   NA
## beta6 -0.44077361  0.08271013  0.67487227  0.08950545 0.28350013   NA
## beta7 -0.31034009  0.04244541  0.37173213  0.04256299 0.17347832   NA
## sigma  0.69006164  0.72142424  0.75342096  0.72145995 0.01616366   NA
##              MCerr MC%ofSD SSeff       AC.200 psrf
## beta0 0.0093617177     4.1   590  0.095231969   NA
## beta1 0.0007975010     4.0   616  0.083523145   NA
## beta2 0.0020741456     2.0  2541  0.005204065   NA
## beta3 0.0010482660     1.4  5000  0.009144817   NA
## beta4 0.0038080208     1.4  5000  0.001185663   NA
## beta5 0.0016691781     1.4  5000 -0.026020203   NA
## beta6 0.0040092973     1.4  5000  0.004342214   NA
## beta7 0.0024533540     1.4  5000  0.004259465   NA
## sigma 0.0002285887     1.4  5000  0.010254110   NA
```

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

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




```
## 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$x_income * post[index, "beta1"] +  X$x_rural * post[index, "beta2
  synthetic_Y <- rnorm(n, mean_Y, post[index, "sigma"])
  data.frame(X$x_income, synthetic_Y)
}
```

i. Generate m = 20 synthetic datasets given your synthesis model for the CE sample. If you are using set.seed(), make sure that you do not generate the same synthetic data for each m = 20.

```
set.seed(123)
m <- 20
n <- dim(origdata)[1]
synthetic_m <- vector("list",m)
new <- data.frame(x_income, x_rural, x_race_B, x_race_N, x_race_A, x_race_P, x_race_M)
for (l in 1:m){
  synthetic_one <- synthesize(new, 4980+l, n)
  names(synthetic_one) <- c("OrigLogIncome", "SynLogIncome")
  synthetic_m[[l]] <- synthetic_one
}
```

ii. Estimate a few analysis-specific utility measures, e.g. the mean and median of a continuous synthetic variable, the regression analysis coefficients, for each synthetic dataset.

```
## Estimates the mean, median, mode, variance, and range of synthetic log Income, as well as regression
mean <- c()
```

```r
median <- c()
mode <- c()
variance <- c()
range <- c()

for (l in 1:m){
  mean[l] = mean(synthetic_m[[l]]$SynLogIncome)
  median[l] = median(synthetic_m[[l]]$SynLogIncome)
  mode[l] = mode(synthetic_m[[l]]$SynLogIncome)
  variance[l] = var(synthetic_m[[l]]$SynLogIncome)
  range[l] = range(synthetic_m[[l]]$SynLogIncome)
  print(lm(origdata$LogExp ~ synthetic_m[[l]]$SynLogIncome))
}
```

```
## Warning in range[l] <- range(synthetic_m[[l]]$SynLogIncome): number of
## items to replace is not a multiple of replacement length

##
## Call:
## lm(formula = origdata$LogExp ~ synthetic_m[[l]]$SynLogIncome)
##
## Coefficients:
##                   (Intercept)  synthetic_m[[l]]$SynLogIncome
##                        5.9110                         0.3257

## Warning in range[l] <- range(synthetic_m[[l]]$SynLogIncome): number of
## items to replace is not a multiple of replacement length

##
## Call:
## lm(formula = origdata$LogExp ~ synthetic_m[[l]]$SynLogIncome)
##
## Coefficients:
##                   (Intercept)  synthetic_m[[l]]$SynLogIncome
##                        6.0121                         0.3145

## Warning in range[l] <- range(synthetic_m[[l]]$SynLogIncome): number of
## items to replace is not a multiple of replacement length

##
## Call:
## lm(formula = origdata$LogExp ~ synthetic_m[[l]]$SynLogIncome)
##
## Coefficients:
##                   (Intercept)  synthetic_m[[l]]$SynLogIncome
##                        6.1379                         0.3024

## Warning in range[l] <- range(synthetic_m[[l]]$SynLogIncome): number of
## items to replace is not a multiple of replacement length

##
## Call:
## lm(formula = origdata$LogExp ~ synthetic_m[[l]]$SynLogIncome)
##
## Coefficients:
##                   (Intercept)  synthetic_m[[l]]$SynLogIncome
##                        5.9313                         0.3243
```

```
## Warning in range[l] <- range(synthetic_m[[l]]$SynLogIncome): number of
## items to replace is not a multiple of replacement length

##
## Call:
## lm(formula = origdata$LogExp ~ synthetic_m[[l]]$SynLogIncome)
##
## Coefficients:
##                  (Intercept)  synthetic_m[[l]]$SynLogIncome
##                       5.6249                         0.3608

## Warning in range[l] <- range(synthetic_m[[l]]$SynLogIncome): number of
## items to replace is not a multiple of replacement length

##
## Call:
## lm(formula = origdata$LogExp ~ synthetic_m[[l]]$SynLogIncome)
##
## Coefficients:
##                  (Intercept)  synthetic_m[[l]]$SynLogIncome
##                       5.8679                         0.3327

## Warning in range[l] <- range(synthetic_m[[l]]$SynLogIncome): number of
## items to replace is not a multiple of replacement length

##
## Call:
## lm(formula = origdata$LogExp ~ synthetic_m[[l]]$SynLogIncome)
##
## Coefficients:
##                  (Intercept)  synthetic_m[[l]]$SynLogIncome
##                        5.803                          0.341

## Warning in range[l] <- range(synthetic_m[[l]]$SynLogIncome): number of
## items to replace is not a multiple of replacement length

##
## Call:
## lm(formula = origdata$LogExp ~ synthetic_m[[l]]$SynLogIncome)
##
## Coefficients:
##                  (Intercept)  synthetic_m[[l]]$SynLogIncome
##                       5.5808                         0.3646

## Warning in range[l] <- range(synthetic_m[[l]]$SynLogIncome): number of
## items to replace is not a multiple of replacement length

##
## Call:
## lm(formula = origdata$LogExp ~ synthetic_m[[l]]$SynLogIncome)
##
## Coefficients:
##                  (Intercept)  synthetic_m[[l]]$SynLogIncome
##                       5.9365                         0.3238

## Warning in range[l] <- range(synthetic_m[[l]]$SynLogIncome): number of
## items to replace is not a multiple of replacement length

##
```

```
## Call:
## lm(formula = origdata$LogExp ~ synthetic_m[[l]]$SynLogIncome)
##
## Coefficients:
##                  (Intercept)  synthetic_m[[l]]$SynLogIncome
##                       6.2384                         0.2919
```

```
## Warning in range[l] <- range(synthetic_m[[l]]$SynLogIncome): number of
## items to replace is not a multiple of replacement length
```

```
##
## Call:
## lm(formula = origdata$LogExp ~ synthetic_m[[l]]$SynLogIncome)
##
## Coefficients:
##                  (Intercept)  synthetic_m[[l]]$SynLogIncome
##                       5.8536                         0.3319
```

```
## Warning in range[l] <- range(synthetic_m[[l]]$SynLogIncome): number of
## items to replace is not a multiple of replacement length
```

```
##
## Call:
## lm(formula = origdata$LogExp ~ synthetic_m[[l]]$SynLogIncome)
##
## Coefficients:
##                  (Intercept)  synthetic_m[[l]]$SynLogIncome
##                       6.2642                         0.2872
```

```
## Warning in range[l] <- range(synthetic_m[[l]]$SynLogIncome): number of
## items to replace is not a multiple of replacement length
```

```
##
## Call:
## lm(formula = origdata$LogExp ~ synthetic_m[[l]]$SynLogIncome)
##
## Coefficients:
##                  (Intercept)  synthetic_m[[l]]$SynLogIncome
##                       5.7997                         0.3395
```

```
## Warning in range[l] <- range(synthetic_m[[l]]$SynLogIncome): number of
## items to replace is not a multiple of replacement length
```

```
##
## Call:
## lm(formula = origdata$LogExp ~ synthetic_m[[l]]$SynLogIncome)
##
## Coefficients:
##                  (Intercept)  synthetic_m[[l]]$SynLogIncome
##                       5.9573                         0.3209
```

```
## Warning in range[l] <- range(synthetic_m[[l]]$SynLogIncome): number of
## items to replace is not a multiple of replacement length
```

```
##
## Call:
## lm(formula = origdata$LogExp ~ synthetic_m[[l]]$SynLogIncome)
##
## Coefficients:
```

```
##                    (Intercept)  synthetic_m[[l]]$SynLogIncome
##                         5.8145                         0.3377

## Warning in range[l] <- range(synthetic_m[[l]]$SynLogIncome): number of
## items to replace is not a multiple of replacement length

##
## Call:
## lm(formula = origdata$LogExp ~ synthetic_m[[l]]$SynLogIncome)
##
## Coefficients:
##                    (Intercept)  synthetic_m[[l]]$SynLogIncome
##                         6.0456                         0.3119

## Warning in range[l] <- range(synthetic_m[[l]]$SynLogIncome): number of
## items to replace is not a multiple of replacement length

##
## Call:
## lm(formula = origdata$LogExp ~ synthetic_m[[l]]$SynLogIncome)
##
## Coefficients:
##                    (Intercept)  synthetic_m[[l]]$SynLogIncome
##                         5.8341                         0.3356

## Warning in range[l] <- range(synthetic_m[[l]]$SynLogIncome): number of
## items to replace is not a multiple of replacement length

##
## Call:
## lm(formula = origdata$LogExp ~ synthetic_m[[l]]$SynLogIncome)
##
## Coefficients:
##                    (Intercept)  synthetic_m[[l]]$SynLogIncome
##                          5.666                          0.356

## Warning in range[l] <- range(synthetic_m[[l]]$SynLogIncome): number of
## items to replace is not a multiple of replacement length

##
## Call:
## lm(formula = origdata$LogExp ~ synthetic_m[[l]]$SynLogIncome)
##
## Coefficients:
##                    (Intercept)  synthetic_m[[l]]$SynLogIncome
##                         5.7171                         0.3501

## Warning in range[l] <- range(synthetic_m[[l]]$SynLogIncome): number of
## items to replace is not a multiple of replacement length

##
## Call:
## lm(formula = origdata$LogExp ~ synthetic_m[[l]]$SynLogIncome)
##
## Coefficients:
##                    (Intercept)  synthetic_m[[l]]$SynLogIncome
##                         5.9466                         0.3232
```

```r
syndata <- synthetic_m[[1]]
```

Step 1: calculate key quantities

```r
known.vars <- c("Rural", "Race", "Expenditure")
syn.vars <- c("LogIncome")
CEdata <- data.frame(origdata$Rural, origdata$Race, origdata$Expenditure, origdata$LogIncome)
CEdatasyn <- data.frame(syndata$Rural, syndata$Race, syndata$Expenditure, syndata$LogIncome)
n <- dim(origdata)[1]
KeyQuantities1 <- CalculateKeyQuantities(CEdata, CEdatasyn, 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)
}

## each record is a target, therefore N = n

c_vector <- KeyQuantities1[["c_vector"]]
T_vector <- KeyQuantities1[["T_vector"]]
K_vector <- KeyQuantities1[["K_vector"]]
F_vector <- KeyQuantities1[["F_vector"]]
s <- KeyQuantities1[["s"]]
N <- n
ThreeSummaries <- IdentificationRisk(c_vector, T_vector, K_vector, F_vector, s, N)
```

Summaries:

```r
## Expected match risk
ThreeSummaries[["exp_match_risk"]]
```

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

```r
## True match rate
ThreeSummaries[["true_match_rate"]]
```

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

```r
## False match rate
ThreeSummaries[["false_match_rate"]]
```

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

Results and Discussion

I could not get the code to work for the identification disclosure risk for the continuous variable.