Disclosure Risk Evaluation #2

Reese Guo

3/22/2020

Identification Risk Evaluation for Categorical Data

```
ACSdata_org <- read.csv(file = "ACSdata_org.csv")
ACSdata_syn1 <- read.csv(file = "ACSdata_syn.csv")
ACSdata_syn2 <- read.csv(file = "ACSdata_syn2.csv")
ACSdata_syn3 <- read.csv(file = "ACSdata_syn3.csv")
```

```
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(origdata)[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", "RACE", "MAR")
syn.vars <- c("LANX", "WAOB", "DIS", "HICOV")
n <- dim(ACSdata_org)[1]
KeyQuantities1 <- CalculateKeyQuantities(ACSdata_org, ACSdata_syn1,
                                        known.vars, syn.vars, n)
KeyQuantities2 <- CalculateKeyQuantities(ACSdata_org, ACSdata_syn2,
                                        known.vars, syn.vars, n)
```

```
KeyQuantities3 <- CalculateKeyQuantities(ACSdata_org, ACSdata_syn3,
                                          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)
}
```

```
c_vector1 <- KeyQuantities1[["c_vector"]]
T_vector1 <- KeyQuantities1[["T_vector"]]
K_vector1 <- KeyQuantities1[["K_vector"]]
F_vector1 <- KeyQuantities1[["F_vector"]]
s1 <- KeyQuantities1[["s"]]
N <- n
ThreeSummaries1 <- IdentificationRisk(c_vector1, T_vector1, K_vector1, F_vector1, s1, N)

c_vector2 <- KeyQuantities2[["c_vector"]]
T_vector2 <- KeyQuantities2[["T_vector"]]
K_vector2 <- KeyQuantities2[["K_vector"]]
F_vector2 <- KeyQuantities2[["F_vector"]]
s2 <- KeyQuantities2[["s"]]
N <- n
ThreeSummaries2 <- IdentificationRisk(c_vector2, T_vector2, K_vector2, F_vector2, s2, N)

c_vector3 <- KeyQuantities3[["c_vector"]]
T_vector3 <- KeyQuantities3[["T_vector"]]
K_vector3 <- KeyQuantities3[["K_vector"]]
F_vector3 <- KeyQuantities3[["F_vector"]]
s3 <- KeyQuantities3[["s"]]
N <- n
ThreeSummaries3 <- IdentificationRisk(c_vector3, T_vector3, K_vector3, F_vector3, s3, N)
```

```
(ThreeSummaries1[["exp_match_risk"]] + ThreeSummaries2[["exp_match_risk"]] + ThreeSummaries3[["exp_matc
```

```
## [1] 41.46743
```

```
(ThreeSummaries1[["true_match_rate"]] + ThreeSummaries2[["true_match_rate"]] + ThreeSummaries3[["true_ma
```

```
## [1] 0.0005666667
```

```
(ThreeSummaries1[["false_match_rate"]] + ThreeSummaries2[["false_match_rate"]] + ThreeSummaries3[["false
```

```
## [1] 0.9638026
```

```
41.46743 / 10000
```

```
## [1] 0.004146743
```

```
0.0005666667 * 10000
```

```
## [1] 5.666667
```

```r
(s1 + s2 + s3) / 3
```

```
## [1] 161
```

```r
(s1 * ThreeSummaries1[["false_match_rate"]] + s2 * ThreeSummaries2[["false_match_rate"]] + s3 * ThreeSu
```

```
## [1] 155.3333
```

The expected match rate 0.004146 is the average probability that a record will be identified correctly. The 0.00056667 true match rate means that 5 records are correct matches. The average amount of unique records of the three synthesized datasets is 161. The false match rate suggests that averagely among 161 unique matches, 155 are false matches.

Identification Disclosure Risk for Continuous Data (CEData)

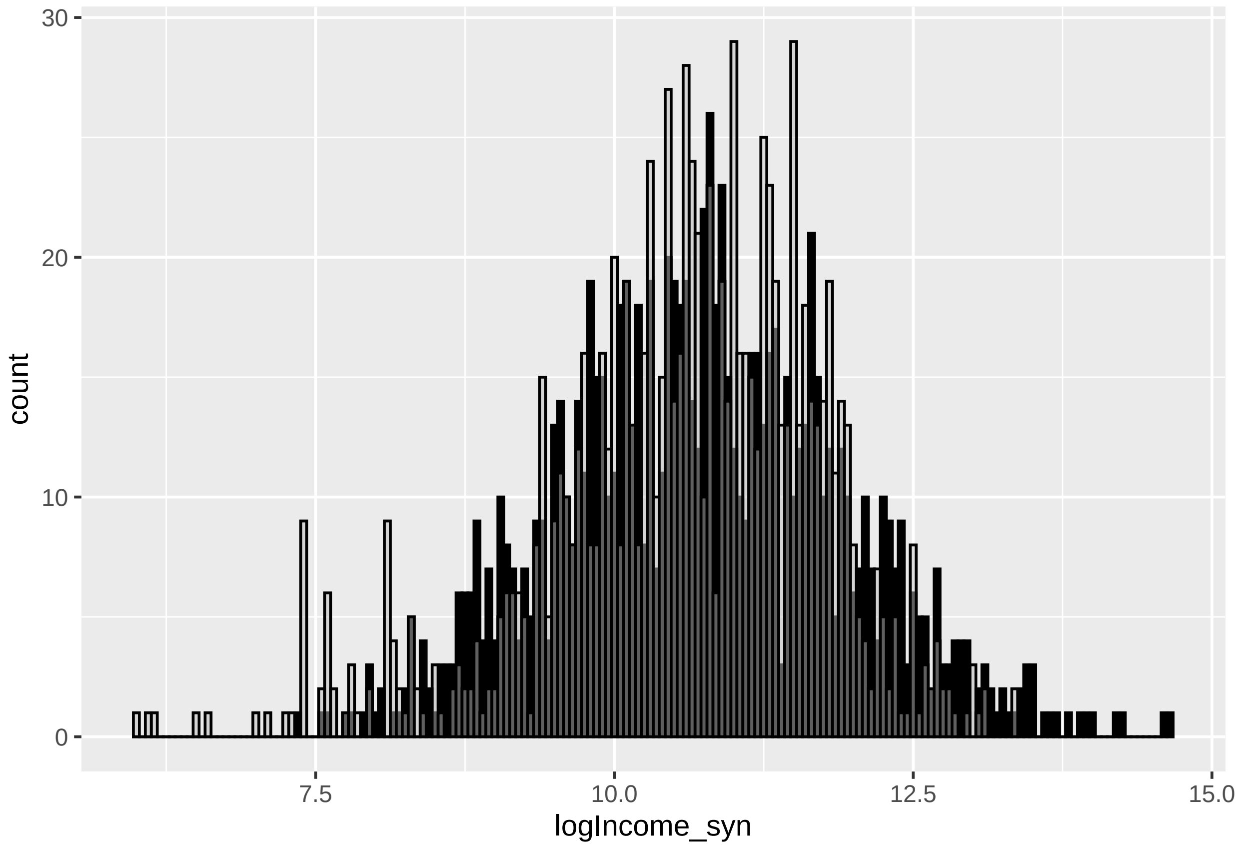
```r
load("March23.RData")
```

```r
logincome_syn <- syn_data$logIncome_syn
logincome_original <- CEdata$LogIncome
Data <- data.frame(syn = logIncome_syn, ori = logincome_original)
```

```r
library(ggplot2)
summary(Data$syn)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   7.372   9.887  10.715  10.723  11.555  14.646
```

```r
ggplot(Data) +
    geom_histogram(aes(x = logIncome_syn), binwidth = 0.05, fill = "black", color = "black") +
    geom_histogram(aes(x = logincome_original),
                   binwidth = 0.05, fill = "grey", color = "black", alpha=0.5)
```




```
length <- length(logIncome_syn)
TF <- c()

for (i in 1:length){
  if( abs(Data$syn[i] - Data$ori) < 0.1)
    TF <- c(TF, 1)
  else
    TF <- c(TF, 0)
}

sum(TF) / length
```

```
## [1] 0.04929577
```

```
summary(logExpenditure)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   4.704   8.206   8.810   8.784   9.370  11.179
```

```
Expen_Cate <- c()
L <- length(logExpenditure)
for(i in 1:L){
  if(logExpenditure[i] < 8.206)
    Expen_Cate <- c(Expen_Cate, 1)
  else if(logExpenditure[i] > 8.206 & logExpenditure[i] < 8.810)
    Expen_Cate <- c(Expen_Cate, 2)
  else if(logExpenditure[i] > 8.810 & logExpenditure[i] < 9.370)
    Expen_Cate <- c(Expen_Cate, 3)
```

```r
  else
    Expen_Cate <- c(Expen_Cate, 4)
}
```

```r
Data_syn <- data.frame(Rural = Rural, Race = Race, Expenditure = Expen_Cate, LogIncome = Data$syn)
Data_ori <- data.frame(Rural = Rural, Race = Race, Expenditure = Expen_Cate, LogIncome = Data$ori)
```

```r
CalculateKeyQuantities_Continuous <- 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("abs(origdata$",syn.vars,"[i] -
            syndata$",syn.vars,") < 0.1",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(origdata)[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)
}
```

```r
known.vars_continuous <- c("Rural", "Race", "Expenditure")
syn.vars_continuous <- c("LogIncome")
n_continuous <- dim(Data_ori)[1]
KeyQuantities_continuous <- CalculateKeyQuantities_Continuous(Data_ori, Data_syn,
                                    known.vars_continuous, syn.vars_continuous, n_continuous)
```

```r
c_vector_cont <- KeyQuantities_continuous[["c_vector"]]
T_vector_cont <- KeyQuantities_continuous[["T_vector"]]
K_vector_cont <- KeyQuantities_continuous[["K_vector"]]
F_vector_cont <- KeyQuantities_continuous[["F_vector"]]
s_cont <- KeyQuantities_continuous[["s"]]
N <- n_continuous
ThreeSummaries_cont <- IdentificationRisk(c_vector_cont, T_vector_cont, K_vector_cont, F_vector_cont, s_
```

```r
ThreeSummaries_cont[["exp_match_risk"]]
```

```
## [1] 5.652312
```

```r
ThreeSummaries_cont[["true_match_rate"]]
```

```
## [1] 0.002012072
```

```r
ThreeSummaries_cont[["false_match_rate"]]
```

```
## [1] 0.96875
```

```r
ThreeSummaries_cont[["exp_match_risk"]] / N
```

```
## [1] 0.005686431
```

```r
ThreeSummaries_cont[["true_match_rate"]] * N
```

```
## [1] 2
```

```r
ThreeSummaries_cont[["false_match_rate"]] * s_cont
```

```
## [1] 62
```

```r
s_cont
```

```
## [1] 64
```

Project Synthesize Methods

```r
bnbData <- read.csv("AB_NYC_2019.csv")
bnbLength <- dim(bnbData)[1]
avail <- bnbData$availability_365
Category <- c()

for(i in 1:bnbLength){
  if (avail[i] > 330){
    Category <- c(Category, 1)
  }else if( avail[i] <= 330 & avail[i] > 270){
    Category <- c(Category, 2)
  }else if( avail[i] <= 270 & avail[i] > 60){
    Category <- c(Category, 3)
  }else{
    Category <- c(Category, 4)
  }
}

room_type <- bnbData$room_type
room_category <- c()

for(i in 1:bnbLength){
  if (room_type[i] == "Private room"){
    room_category <- c(room_category, 1)
  }else if( room_type[i] == "Entire home/apt"){
    room_category <- c(room_category, 2)
  }else{
    room_category <- c(room_category, 3)
```

```
  }
}

neigh <- bnbData$neighbourhood_group
neigh_category <- c()

for(i in 1:bnbLength){
  if (neigh[i] == "Brooklyn"){
    neigh_category <- c(neigh_category, 1)
  }else if(neigh[i] == "Manhattan"){
    neigh_category <- c(neigh_category, 2)
  }else if(neigh[i] == "Queens"){
    neigh_category <- c(neigh_category, 3)
  }else if(neigh[i] == "Staten Island"){
    neigh_category <- c(neigh_category, 4)
  }else{
    neigh_category <- c(neigh_category, 5)
  }
}

bnbData_cat <- data.frame(bnbData, category = Category, room_category = room_category, neigh_category =
neigh_unique <- unique(bnbData_cat$neighbourhood_group)
room_unique <- unique(bnbData_cat$room_type)
price <- bnbData$price
```

```
modelString <-"
model {
  ## sampling
  for (i in 1:n){
    ez<- t (beta)%*%X[i, ]
    a<-max(-Inf , g[y[i] -1] , na.rm=TRUE)
    b<-min( g[y[i]] , Inf , na.rm=TRUE)
    u<-runif(1 , pnorm( a-ez ) , pnorm(b-ez ) )
    z[i]<- ez + qnorm(u)
  }
  for (k in 1:K){
    c<-max( z [ y==k ] )
    d<-min( z [ y==k+1])
    u<-runif ( 1 , pnorm( ( c-mu[k] ) / sig[k] ) , pnorm( ( d-mu[k] ) / sig[k] ) )
    g [k]<- mu[k] + sig[k] * qnorm(u)
  }
  ## priors
  beta ~ dnorm(n/((n+1) * (t(X)%*%X)) %*% t(X) %*% z, n/((n+1) * (t(X)%*%X)))
  for(k in 1:K){
    mu[k] ~ dnorm(0, 1)
    sig[k] ~ dnorm(0, 1)
  }
}"
```

```
X <- cbind(room_category, neigh_category, price)
y <- bnbData_cat$category
n <- dim(x)[2]
K <- 4
```

```r
the_data <- list("X" = X, "y" = y, "n" = n, "K" = K)

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))
}

require(runjags)

posterior <- run.jags(modelString,
                      n.chains = 1,
                      data = the_data,
                      monitor = c("z", "beta", "g"),
                      adapt = 1000,
                      burnin = 5000,
                      sample = 5000,
                      thin = 1,
                      inits = initsfunction)
```

```r
# full conditional for beta
beta ~ dnorm(n/((n+1) * (t(X)%*%X)) %*% t(X) %*% z, n/((n+1) * (t(X)%*%X)))
# full conditional for Z
ez<- t ( beta)%*%X[ i , ]
a<-max(-Inf , g [ y [ i ] -1] , na.rm=TRUE)
b<-min( g [ y [ i ] ] , Inf , na.rm=TRUE)
u<-runif ( 1 , pnorm( a-ez ) , pnorm(b-ez ) )
z [ i ]<- ez + qnorm(u)
# full conditional for g
a<-max( z [ y==k ] )
b<-min( z [ y==k+1])
u<-runif ( 1 , pnorm( ( a-mu[ k ] ) / sig[k] ) , pnorm( ( b-mu[k] ) / sig[k] ) )
g [k]<- mu[k] + sig[k] * qnorm(u)
```