Interval-overlap

Reese Guo

3/2/2020

```
load("Utility-Combine_Rules.RData")
```

Overlay Interval

```
syn_data <- synthetic_m[[1]]
logincome_syn <- syn_data$logIncome_syn
logincome_original <- CEdata$LogIncome
L_s <- unname(quantile(logincome_syn, 0.025))
U_s <- unname(quantile(logincome_syn, 0.975))
L_o <- unname(quantile(logincome_original, 0.025))
U_o <- unname(quantile(logincome_original, 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
```

```
## [1] 0.5146028
```

Since data with high utility would have a score of 1 and no data with no utility will have a score of 0, our synthesized data has a medium utility.

Expected Match Risk

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
ACS_ori <- read.csv("ACSdata_org.csv")
ACS_syn <- read.csv("ACSdata_syn.csv")

Sex_syn <- ACS_syn$SEX
Race_syn <- ACS_syn$RACE
Mar_syn <- ACS_syn$MAR

data <- cbind(ACS_ori, SEX_syn = Sex_syn, RACE_syn = Race_syn, MAR_syn = Mar_syn)
N <- dim(data)
row_num <- as.vector(1:N)
```

```r
ACS_ori <- cbind(ACS_ori, row = row_num)
ACS_syn <- cbind(ACS_syn, row = row_num)
EMR <- c()

expect_match_risk <- function(x){
  s <- Sex_syn[x]
  r <- Race_syn[x]
  m <- Mar_syn[x]
  select <- which(ACS_ori$SEX == s & ACS_ori$RACE == r & ACS_ori$MAR == m)
  c_i <- length(select)
  if(x %in% select){
    return (1/c_i)
  }
  else{
    return (0)
  }
}

for(i in 1:N){
  EMR <- c(EMR, expect_match_risk(i))
}
```

```r
summary(EMR)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0004562 0.0004581 0.0012422 0.0057000 0.0026247 1.0000000
```

True Match Rate

```r
true_match_rate <- function(x){
  s <- Sex_syn[x]
  r <- Race_syn[x]
  m <- Mar_syn[x]
  select <- which(ACS_ori$SEX == s & ACS_ori$RACE == r & ACS_ori$MAR == m)
  c_i <- length(select)
  if(x %in% select & c_i == 1){
    return (1/N)
  }
  else{
    return (0)
  }
}

TMR <- c()

for(i in 1:N){
  TMR <- c(TMR, true_match_rate(i))
}

summary(TMR)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.000e+00 0.000e+00 0.000e+00 3.336e-05 0.000e+00 8.333e-02
```

False Match Rate

```
compute_c_i <- function(x){
  s <- Sex_syn[x]
  r <- Race_syn[x]
  m <- Mar_syn[x]
  select <- which(ACS_ori$SEX == s & ACS_ori$RACE == r & ACS_ori$MAR == m)
  c_i <- length(select)
  return (c_i)
}

c_i <- c()

for(i in 1:N){
  c_i <- c(c_i, compute_c_i(i))
}

ACS_ori <- cbind(ACS_ori, c_i = c_i)
```

```
data_filter <- filter(ACS_ori, c_i == 1)
s_fms <- dim(data_filter)

false_match_rate <- function(x){
  s <- Sex_syn[x]
  r <- Race_syn[x]
  m <- Mar_syn[x]
  select <- which(ACS_ori$SEX == s & ACS_ori$RACE == r & ACS_ori$MAR == m)
  c_i <- length(select)
  if((!x %in% select) & c_i == 1){
    return (1/s_fms)
  }
  else{
    return (0)
  }
}

FMS <- c()

for(i in 1:N){
  FMS <- c(FMS, false_match_rate(i))
}
```

```
summary(FMS)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0       0       0       0       0       0
```

Project Synthesize Method

```
bnbData <- read.csv("AB_NYC_2019.csv")
length <- dim(bnbData)
avail <- bnbData$availability_365
Category <- c()
```

```r
for(i in 1:length){
  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)
  }
}
```

```
## Warning in 1:length: numerical expression has 2 elements: only the first
## used
```

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

for(i in 1:length){
  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)
  }
}
```

```
## Warning in 1:length: numerical expression has 2 elements: only the first
## used
```

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

for(i in 1:length){
  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)
  }
}
```

```
## Warning in 1:length: numerical expression has 2 elements: only the first
## used
```

```r
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_alt <-"
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)
  a<-max( z[y==k] )
  b<-min( z[y==k+1])
  sig[i] ~ dnorm(0,1)
  u<-runif(1 , pnorm( ( a - ez ) / sig[i] ) , pnorm(( b - ez ) / sig[i] ))
  g[i]<- ez + sig[i] * qnorm(u)
}
beta ~ dnorm(n/((n+1) * (t(X)%*%X)) %*% t(X) %*% z, n/((n+1) * (t(X)%*%X)))
}"
```

```
X <- cbind(room_category, neigh_category, price)
y <- bnbData_cat$category

the_data <- list("X" = X, "y" = y)

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_alt,
                      n.chains = 1,
                      data = the_data,
                      monitor = c("z", "beta", "g"),
                      adapt = 1000,
                      burnin = 5000,
                      sample = 5000,
                      thin = 50,
                      inits = initsfunction)
```