Methods for Risk Evaluation #2

Jingchen (Monika) Hu

Vassar College

Data Confidentiality

Outline

Outline

Recap of $m = 1$

- A good practice to write functions to calculate various quantities
- When `m > 1`
  - create `c_vector`, `T_vector`, `K_vector`, `F_vector` as matrices $(n - by - m)$
  - create `s` as a vector of length m
  - add nested loops when necessary
  - create `exp_match_risk`, `true_match_rate`, `false_match_rate` as vectors of length m
  - `syndata` is a list

ACS sample information

Variable	Information
SEX	1 = male, 2 = female
RACE	1 = White alone, 2 = Black or African American alone, 3 = American Indian alone, 4 = other, 5 = two or more races, 6 = Asian alone
MAR	1 = married, 2 = widowed, 3 = divorced, 4 = separated, 5 = never married
LANX	1 = speaks another language, 2 = speaks only English
WAOB	born in: 1 = US state, 2 = Puerto Rico and US island areas, oceania and at sea, 3 = Latin America, 4 = Asia, 5 = Europe, 6 = Africa, 7 = Northern America
DIS	1 = has a disability, 2 = no disability
HICOV	1 = has health insurance coverage, 2 = no coverage
MIG	1 = live in the same house (non movers), 2 = move to outside US and Puerto Rico, 3 = move to different house in US or Puerto Rico
SCH	1 = has not attended school in the last 3 months, 2 = in public school or college, 3 = in private school or college or home school

ACS sample information cont'd

- `ACSdata_org`: the original ACS sample
- `ACSdata_syn`, `ACSdata_syn2`, and `ACSdata_syn3`: three synthetic ACS samples
    - four variables are synthesized: `LANX, WAOB, DIS, HICOV`
    - `m = 3`

ACS sample information cont'd

- `ACSdata_org`: the original ACS sample
- `ACSdata_syn`, `ACSdata_syn2`, and `ACSdata_syn3`: three synthetic ACS samples
  - four variables are synthesized: `LANX`, `WAOB`, `DIS`, `HICOV`
  - `m = 3`
- Known variables: `SEX`, `RACE`, `MAR`
- Goal: use this information to identify records in `ACSdata_syn`, obtain the 3 summaries

```
ACSdata_org <- read.csv(file = "ACSdata_org.csv")
ACSdata_syn <- read.csv(file = "ACSdata_syn.csv")
ACSdata_syn2 <- read.csv(file = "ACSdata_syn2.csv")
ACSdata_syn3 <- read.csv(file = "ACSdata_syn3.csv")
ACSdata_syn_all <- list(ACSdata_syn, ACSdata_syn2, ACSdata_syn3)
```

The IdentificationRisk function - page 1

```
IdentificationRisk <- function(origdata, syndata, known.vars, syn.vars,
                               m, n){
  origdata <- origdata
  syndata <- syndata
  m <- m
  n <- n

  c_vector <- matrix(rep(NA, n*m), ncol = m)
  T_vector <- matrix(rep(NA, n*m), ncol = m)
```

The IdentificationRisk function - page 2

```
  for (i in 1:n){
    for (k in 1:m){
      syndata_k <- syndata[[k]]
      match_k <- (eval(parse(text=paste("origdata$",syn.vars,"[i]==
                                        syndata_k$",syn.vars,sep="",
                                        collapse="&")))&
                  eval(parse(text=paste("origdata$",known.vars,"[i]==
                                          syndata_k$",known.vars,sep="",
                                          collapse="&"))))
      match.prob_k <- ifelse(match_k,1/sum(match_k),0)

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

The IdentificationRisk function - page 3

```
K_vector <- matrix(rep(NA, n*m), ncol = m)
F_vector <- matrix(rep(NA, n*m), ncol = m)
for (k in 1:m){
  K_vector[, k] <- (c_vector[, k]*T_vector[, k]==1)
  F_vector[, k] <- (c_vector[, k]*(1 - T_vector[, k])==1)
}

s_vector <- rep(NA, m)
exp_match_risk_vector <- rep(NA, m)
true_match_rate_vector <- rep(NA, m)
false_match_rate_vector <- rep(NA, m)

for (k in 1:m){
  s_vector[k] <- length(c_vector[c_vector[, k]==1 &
                                  is.na(c_vector[, k])==FALSE, k])
  nonzero_c_index <- which(c_vector[, k]>0)
  exp_match_risk_vector[k] <- sum(1/c_vector[nonzero_c_index, k]
                              * T_vector[nonzero_c_index, k])
  true_match_rate_vector[k] <- sum(na.omit(K_vector[, k]))/n
  false_match_rate_vector[k] <- sum(na.omit(F_vector[, k]))/s_vector[k]
}
```

The IdentificationRisk function - page 4

```
  res_r <- list(s_vector = s_vector,
                exp_match_risk_vector = exp_match_risk_vector,
                true_match_rate_vector = true_match_rate_vector,
                false_match_rate_vector = false_match_rate_vector,
                c_vector = c_vector,
                T_vector = T_vector,
                K_vector = K_vector,
                F_vector = F_vector
  )
  return(res_r)
}
```

Running the function

```r
known.vars <- c("SEX", "RACE", "MAR")
syn.vars <- c("LANX", "WAOB", "DIS", "HICOV")
n <- dim(ACSdata_org)[1]
m <- length(ACSdata_syn_all)

output <- IdentificationRisk(ACSdata_org, ACSdata_syn_all,
                             known.vars, syn.vars, m, n)
```

Results and discussion

```r
mean(output[["exp_match_risk_vector"]])
```

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

```r
mean(output[["true_match_rate_vector"]])
```

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

```r
mean(output[["false_match_rate_vector"]])
```

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

Results and discussion

- The 41.47 expected match risk: $\frac{41.47}{10000} = 0.000042$ probability on average for each record to be correctly identified

Results and discussion

- The 41.47 expected match risk: $\frac{41.47}{10000} = 0.000042$ probability on average for each record to be correctly identified
- The 0.0006 true match rate: $0.0006 \times 10000 = 6$ records are correct unique matches

Results and discussion

- The 41.47 expected match risk: $\frac{41.47}{10000} = 0.000042$ probability on average for each record to be correctly identified
- The 0.0006 true match rate: $0.0006 \times 10000 = 6$ records are correct unique matches
- The 0.96 false match rate: among the 161 (the value of s) unique matches, about 155 are false matches, i.e. they are not the true matches

```
mean(output[["s_vector"]])
```

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

Results and discussion

- The 41.47 expected match risk: $\frac{41.47}{10000} = 0.000042$ probability on average for each record to be correctly identified
- The 0.0006 true match rate: $0.0006 \times 10000 = 6$ records are correct unique matches
- The 0.96 false match rate: among the 161 (the value of s) unique matches, about 155 are false matches, i.e. they are not the true matches

```
mean(output[["s_vector"]])
```

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

- Overall, the identification disclosure risks for the synthetic ACS sample seem very low, indicating a high level of confidentiality protection of the synthetic ACS data

Outline

What do we need?

- c_i: the number of records with the highest match probability for the target record i.
    1. the records with the highest match probability for record i are a subset of all the records sharing the same known information by the intruder.
    2. e.g. for categorical variable(s), we can consider all records in the same known pattern with record i.
    3. e.g. for continuous variable(s), we can consider all records within a certain distance from record i.
- T_i: if the true match is among the c_i units, $T_i = 1$; otherwise $T_i = 0$.

What do we need? cont'd

- K_i: if the true match is the unique match (i.e. $c_i T_i = 1$), $K_i = 1$; otherwise $K_i = 0$.
- F_i: if there is a unique match but it is not the true match (i.e. $c_i(1 - T_i) = 1$), $F_i = 1$; otherwise $F_i = 0$.
- N: the total number of target records; typically $N = n$, the number of records in the sample.
- s: the number of uniquely matched records (i.e. $\sum_{i=1}^{n} c_i = 1$).

The three summaries

- The expected match risk
  - on average how likely it is to find the correct match for each record, and for the sample as a whole

$$\sum_{i=1}^{n} \frac{T_i}{c_i} \tag{1}$$

The three summaries con'td

- The true match rate
  - how large a percentange of true unique mathces exists

$$\sum_{i=1}^{n} \frac{K_i}{N} \tag{2}$$

- The false match rate
  - the percentage of unique matches to be false matches

$$\sum_{i=1}^{n} \frac{F_i}{s} \tag{3}$$

What are your methods?

Some coding techniques

- Categorical case:

```
match_k <- (eval(parse(text=paste("origdata$",syn.vars,"[i]==
                                   syndata_k$",syn.vars,sep="",
                                   collapse="&")))&
                    eval(parse(text=paste("origdata$",known.vars,"[i]==
                                     syndata_k$",known.vars,sep="",
                                     collapse="&"))))
```

Some coding techniques cont'd

- Continuous case:
  - example of synthesized variables: one univariate continuous, e.g. (`syn.vars <- c("Income")`)
  - `radius`: the distance from the true value of the synthesized univariate continuous value, e.g. `Income`
  - the distance can be an absolute value (e.g. \$500 for every CU) or a percentage (e.g. 20% for every CU)

```
match_k<-(eval(parse(text=paste("origdata$",known.vars,"[i]==
                                        syndata_k$",known.vars,sep="",
                                        collapse="&")))&
                  (eval(parse(text=paste("syndata_k$",syn.vars,"<=
                                      origdata$",syn.vars,"[i]+",
                                      radius,sep="",collapse="&")))&
                     eval(parse(text=paste("syndata_k$",syn.vars,">=
                                      origdata$",syn.vars,"[i]-",
                                      radius,sep="",collapse="&")))))
```

Discussions

- What if we have more than one synthesized continuous variables? How can you create the corresponding `radius`?