SafeDeID

Ng Wei Lin, Lukas Loh

Introduction

Dealing with high volumes of patient identifiers constitutes a core component of a data analyst's daily work in RHSO, NUHS. Data is also typically unsanitised which requires effort to remove after the extraction process. Prior measures are tedious and manual, which exposes the team to risk of breaching PDPA guidelines, especially when analysed data is shared with others. The team's aim is to develop an elegant and comprehensive solution which addresses the weaknesses, thereby reducing time spent on data cleaning and more time can be spent on higher value-adding data analysis.

```
1  library(base)
2  library(MASS)
3  library(tools)
4
5  ##Set working directory. Change ADID and location of csv files
6  rm(list=ls())
7  wd = "C:/Users/yourADID/wherever your saved your csv files"
8  setwd(wd)
9  files <- list.files(path=getwd(), pattern=".csv")
10
11
12 dfList <- lapply(files, function(f) {
13   df <- read.csv(f)
14
15   ### Able to handle different column names for Patient NRIC
16   colnames(df)[colnames(df) == "Patient.Code"] <- "Patient.Code"
17   colnames(df)[colnames(df) == "Patient.MRN"] <- "Patient.Code"
18   colnames(df)[colnames(df) == "Patient.NRIC"] <- "Patient.Code"
19
20   df$Patient.Code=as.character(df$NRIC) # change the inner bracket NRIC accordingly
21
22   ### Ensure that NRIC is valid (len=9)
23   if (length(df$Patient.Code)!=9){
24     cat("removed", sum(nchar(df$Patient.Code) != 9), "row/s.\n")
25     df <- df[!(nchar(df$Patient.Code) != 9),]
26   }
27
28   ### Run deidentification, change location to where deidentification key is located
29   deidentification= readRDS("C:/Users/yourADID/location of deidentification_key.rds")
30   masked=deidentification(unique(df$Patient.Code), reverse=F)
31   names(masked)=c('Patient.Code', 'Masked.Patient.NRIC')
32
33   ###Join masked id to original data and delete NRIC from the original data
34   df2=merge(df, masked, by = 'Patient.Code')
35   df2$Patient.Code=NULL
36
37   ### Output the masked csv files in the same location, labelled as 'masked'
38   write.csv(df2, paste0(wd,'/(masked) ', f), row.names=F)
39   return(df2)
40 })
```

Figure 1: R Script for Deidentification of files containing Patient Identifiable Data

Objective

To develop a programming script in R that is capable to perform the cleaning and masking of patient identifiers for large, multiple datasets in a time efficient way.

Methods

The procedure begins with storing the files to be masked alongside the deidentification key and the R script (Figure 1). These files can either be individual files or organized within folders, depending on the datasets. The working directory in the R script is adjusted accordingly (lines 7-8). If the files to be masked have varying column names for Patient NRIC, modifications are made in the script (lines 16-19) to eliminate the need for manual changes. NRIC validation is then conducted to ensure that they are 9 characters in length (lines 23-26). Subsequently, the NRICs are deidentified using the deidentification key. The masked NRICs are displayed as a new column labelled "Masked Patient NRIC." Following this, the actual NRICs are removed (line 35) before saving the files in the same folder (line 38). Finally, the output file is labelled as masked to prevent confusion with the original dataset containing identifiable data.

Conclusion

In data analysis within data-intensive environments, where privacy and data security are crucial, this innovation stands out for its ability to streamline processes and enhance productivity. This innovation significantly reduces margin of error and boosts efficiency in masking patient identifiers during data cleaning. Its scalability allows for rapid implementation with minimal iterations to adapt to varying column names across databases. This automation not only accelerates the deidentification process but also contributes to workforce transformation by facilitating seamless collaboration and information sharing among analysts and departments. However, The foreseeable challenge for this project would be the feasibility of the solution for databases which are not currently known to the project team. There may also be other types of exceptions and errors which the solution is not able to circumvent in its current state which will require further refinement.