Lab Project

Practical Differential Privacy: Healthcare Scenario

Version 1.1

LERSAIS

School of Computing and Information

University of Pittsburg

This lab is designed by *Runhua Xu* and also part of lab module for SAHI Project

Goal

The goal of this lab project is to illustrate the motivation of differential privacy technique in the healthcare domain and introduce two types of approaches for practical differential privacy: generic mechanisms for differential privacy and sensitivity sampling for random differential privacy.

Tools

To perform this lab, you will need to have the R programming environment installed as we will use a library called ***diffpriv*** which is a package making privacy-aware data science in R platform. It implements the formal framework of differential privacy mechanism.

Part 0: Required Preliminaries

0.1 Setup

Before you start this lab, you need to setup the R programming environment, namely, install R and a related development environment. Here we only provide several links instead of providing the instructions of the installation and setup.

- Official website of R
    - https://www.r-project.org/
- The manuals of R
    - https://cran.r-project.org/doc/manuals/R-admin.html
- A preferred SDE, RStudio
    - https://www.rstudio.com/
        - The free version RStudio is satisfied for your lab
- ***randomNames*** package
    - https://cran.r-project.org/web/packages/randomNames/vignettes/randomNames.html
        - After your setup on R and RStudio, you can install *diffpriv* package via the following command in the console window:

          ```
          install.packages(“randomNames”)
          ```
- ***diffpriv*** package
    - http://www.bipr.net/diffpriv/
        - After your setup on R and RStudio, you can install *diffpriv* package via the following command in the console window:

          ```
          install.packages(“diffpriv”)
          ```

0.2 Preliminary questions

Q0: Answer the following questions:

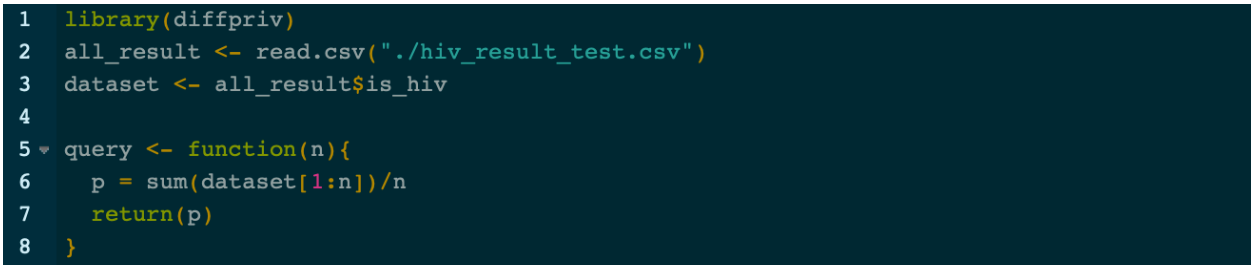
a. In your own words, define what *differential privacy* is
b. In your own words, define what *sensitivity* of a function in the context of differential privacy is
c. Explain the trade-off between utility and privacy in the context of differential privacy.
d. In your own words, define what *random differential privacy* is
e. For differentially private mechanisms, explain the difference between *Laplace* mechanism and *Exponential* mechanism.

Part I: Motivation scenario

Suppose that a hospital cooperates with a research center to do several disease analyses based on the patients' checking result at the hospital.

Take the HIV as an example, the hospital has the examination result of its patients, as presented in the file "hiv_result_test.csv". You can go through the csv file and have an intuition impression of the test result. The examination result includes HIV checking result of 100 patients. Specifically, each row of the csv file indicates one patient examination, including the patient name, HIV examination result (1 indicates having HIV; 0 denotes no HIV).

However, HIV examination result is the sensitive private information of the patients, hence, it is impossible to share the csv file to the cooperator, namely, the research center. As a result, the hospital can just provide query service for the statistical result instead of sharing the whole original dataset or anonymized dataset. For instance, one query function may be as follows:



```r
library(diffpriv)
all_result <- read.csv("./hiv_result_test.csv")
dataset <- all_result$is_hiv

query <- function(n){
  p = sum(dataset[1:n])/n
  return(p)
}
```

The query function *query(n)* computes the percentage of HIV patients in the total patients, where the parameter *n* denotes the coverage of the patients in this query. For example, *query(10)* will returns the percentage of HIV patients among the 10 patients (patient id from 1 to 10). However, such query function has vulnerability that may leak the patients' private information.

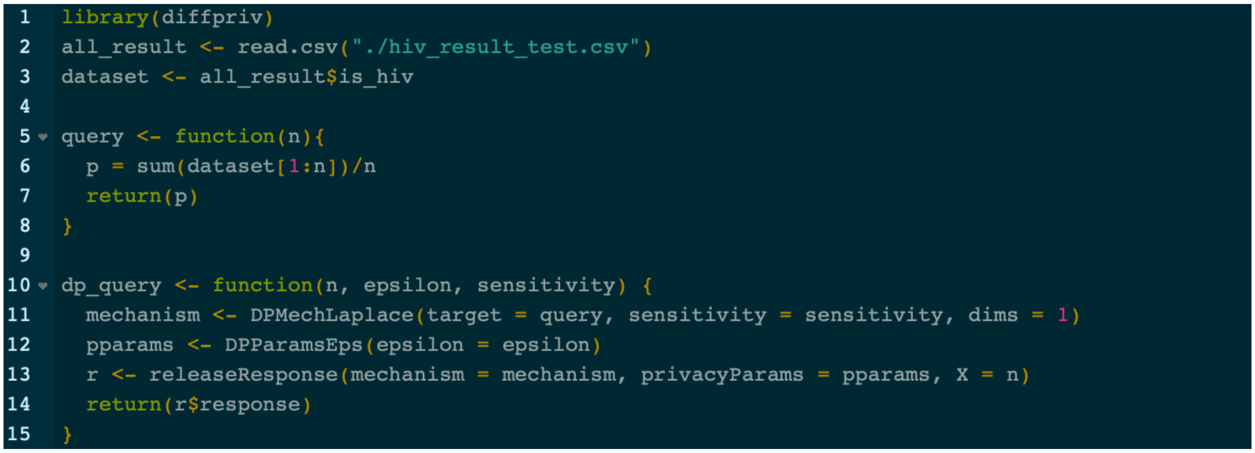
Suppose that you are the adversary who wants to exploit the leakage of the query service, and also a background knowledge such that

1. "The total patients in the examination result was 99 and the order of the examination will not be shuffled after each result appending"
2. "Kyle Winkler went to hospital for HIV examination yesterday, hence, it indicates that he is the 100th patient in the examination result file"

Q1: Based on the information mentioned above, and query service, how do you lunch the attack to identify whether Kyle Winkler has HIV or not? Provide the specific steps and final result. Note that the only approach to access the csv file is implementing the query function to get the query result.

Part II: Generic mechanism for differential privacy

In this part, we will tackle the privacy leakage issue described in the scenario described above. As you have answered in Q1, the previous designed query function has the risk to be exploited by the adversary. Here we designed another query function that implemented the Laplace mechanism, as shown following:



```r
library(diffpriv)
all_result <- read.csv("./hiv_result_test.csv")
dataset <- all_result$is_hiv

query <- function(n){
  p = sum(dataset[1:n])/n
  return(p)
}

dp_query <- function(n, epsilon, sensitivity) {
  mechanism <- DPMechLaplace(target = query, sensitivity = sensitivity, dims = 1)
  pparams <- DPParamsEps(epsilon = epsilon)
  r <- releaseResponse(mechanism = mechanism, privacyParams = pparams, X = n)
  return(r$response)
}
```

Different from previous normal query function, we employed the class DPMechLaplace (Dwork et al., 2006) provided in the diffpriv package that has implemented the Laplace mechanism. The r code has been provided with the simulated dataset, you can load them into your R environment.

Q2: Based on the provided implemented query functions, run the code in your environment and then answer the following questions:

a. Set the variable *n = 99, epsilon = 1, sensitivity = 1/n*, what is the output of the normal query function *query(n)* and ε-differential privacy query function *dp_query(n, epsilon, sensitivity)*?

b. Set the variable *n = 100, epsilon = 1, sensitivity = 1/n*, what is the output of the normal query function *query(n)* and ε-differential privacy query function *dp_query(n, epsilon, sensitivity)*?

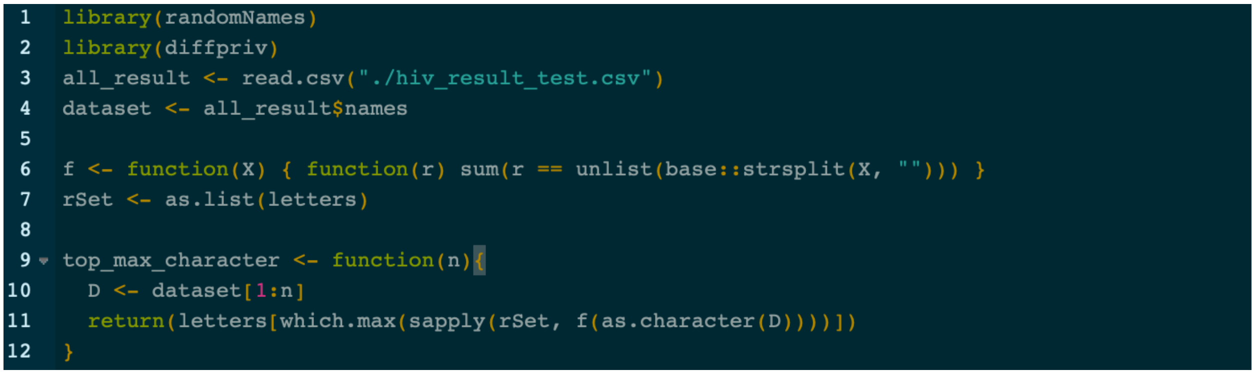
c. Is it possible to lunch the attack as you have done in Q1? Explain the reason using the .

d. Try to use different value for variable *epsilon*, such as 0.01, 0.05, 0.1, 0.5, 1, 10, and analyze the *dp_query* output and compare to the normal *query* output. What is the influence of different *epsilon* settings to the differentially private output, regarding to the privacy and utility?

Part III: Sensitivity Sampling for Random Differential Privacy

As we know the Laplace mechanism is a kind of generic mechanism for differential privacy, in this part, we focus on the sensitivity sampling for random differential privacy to address the case where the global sensitivity of the query function may not be readily available.

Recall the scenario described in Part I. Suppose that we have another query function that is used to find the most frequent a-z character within the dataset of all the names listed in the csv file. (For demo here, just ignore the real purpose of this query function). The details of the function are described as follows:



```r
library(randomNames)
library(diffpriv)
all_result <- read.csv("./hiv_result_test.csv")
dataset <- all_result$names

f <- function(X) { function(r) sum(r == unlist(base::strsplit(X, ""))) }
rSet <- as.list(letters)

top_max_character <- function(n){
  D <- dataset[1:n]
  return(letters[which.max(sapply(rSet, f(as.character(D))))])
}
```

In this part of lab, you will first learn how to exploit the leakage of the query function *top_max_character(n)* and then learn how to use exponential mechanism and sensitively sampling to generate random differential privacy result.

Exploitation: Suppose that ou are the adversary who wants to exploit the leakage of the query service, and also a background knowledge such that

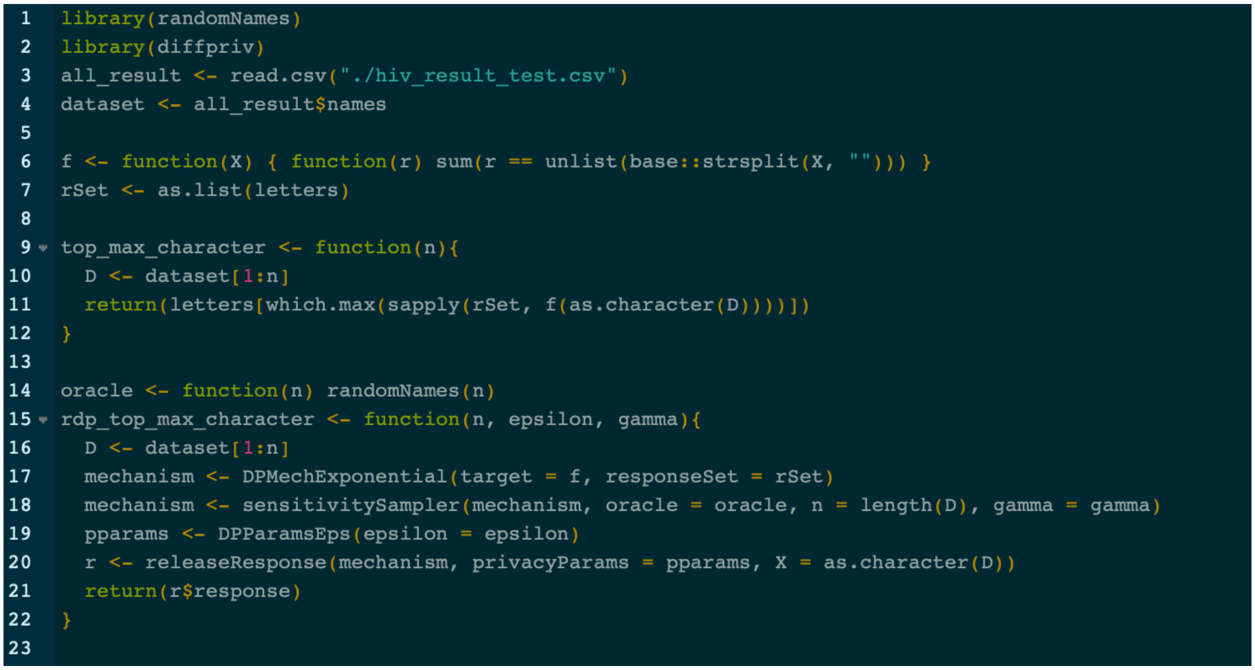
1. “The total patients in the examination result was 100”
2. “Banjamine Khjaa went to hospital for some examinations yesterday”

Q3: Based on the information mentioned above, and top_max_character(n) service, how do you lunch the attack to identify whether Banjamine Khjaa did the HIV examination or not? Provide the specific steps and final result. Note that the only approach to access the csv file is implementing the top_max_character(n) function to get the query result.

Before the mechanism to address the privacy leakage you found in the Q3, here is formal definition of (ε, γ)- random differential privacy:

Definition (Hall et al., 2012) A mechanism M preserves (ε,,γ)-random differential privacy (with a corresponding form for ε, δ, γ) if ∀R ⊆ R, Pr (M(D) ∈ R) ≤ exp(ε) · Pr (M(D') ∈ R) holds with probability at least 1 − γ over random database pairs D, D'.

The random differential privacy based *top_max_character* function is depicted as follows:



```
library(randomNames)
library(diffpriv)
all_result <- read.csv("./hiv_result_test.csv")
dataset <- all_result$names

f <- function(X) { function(r) sum(r == unlist(base::strsplit(X, ""))) }
rSet <- as.list(letters)

top_max_character <- function(n){
  D <- dataset[1:n]
  return(letters[which.max(sapply(rSet, f(as.character(D))))])
}

oracle <- function(n) randomNames(n)
rdp_top_max_character <- function(n, epsilon, gamma){
  D <- dataset[1:n]
  mechanism <- DPMechExponential(target = f, responseSet = rSet)
  mechanism <- sensitivitySampler(mechanism, oracle = oracle, n = length(D), gamma = gamma)
  pparams <- DPParamsEps(epsilon = epsilon)
  r <- releaseResponse(mechanism, privacyParams = pparams, X = as.character(D))
  return(r$response)
}

```

Q4: Based on the provided implemented query functions, run the code in your environment and then answer the following questions:

a. Set the variable *n = 100, epsilon = 1, gamma = 0.2*, what is the output of the normal query function *top_max_character (n)* and (ε,γ)-random differential privacy function *rdp_top_max_character (n, epsilon, gamma)*?

b. Set the variable *n = 101, epsilon = 1, gamma = 0.2*, what is the output of the normal query function *top_max_character (n)* and (ε,γ)-random differential privacy function *rdp_top_max_character (n, epsilon, gamma)*?

c. Is it possible to lunch the attack as you have done in Q3? Explain the reason using the .

d. For setting *epsilon = 1*, try to use different value for variable *gamma*, such as 0.1, 0.2, 0.5, 0.9, and analyze the *rdp_top_max_character (n, epsilon, gamma)* output and compare to the normal *top_max_character (n)* output. What is the influence of different *gamma* settings to the differentially private output, regarding to the privacy and utility?

e. For setting *gamma = 0.2*, try to use different value for variable *epsilon*, such as 0.01, 0.05, 0.1, 0.5, 1, 10, and analyze the *rdp_top_max_character (n, epsilon, gamma)* output and compare to the normal *top_max_character (n)* output. What is the influence of different *epsilon* settings to the differentially private output, regarding to the privacy and utility?

Reference Reading List

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In Theory of Cryptography Conference, pages 265– 284. Springer, 2006.
Rob Hall, Alessandro Rinaldo, and Larry Wasserman. Random differential privacy. Journal of Privacy and Confidentiality, 4(2):43–59, 2012