Privacy-Preserving Data Publishing
Homework : Implementing the Laplace Mechanism

Deliverables : *Please send all deliverables to* `tristan.allard@irisa.fr`.

- *Code : a zip file containing your code (please comment it thoroughly !). No executable. Allowed formats : zip, tar, gz.*
- *Report : a written document that contains your graphs and your comments. 5 pages at most. Allowed formats : pdf.*
- *If you use Python, it is possible to submit your code and report in a single Python notebook file.*

Group : *each group is made of 3 students. If needed, a single group is allowed to contain a different number of students.*

Deadlines : *You must meet the following two deadlines :*

- *Intermediate version : 15^{th} December 2019*
- *Final version : 19^{th} January 2020*

Programming language : *This practical work session is written for programming in Java but if you prefer Python, please do not hesitate to switch to Python.*

Goals

1. Implement the Laplace mechanism for satisfying ϵ-differential privacy
2. Analyze the errors due to the Laplace perturbation
3. Distribute the privacy budget

In this section you are going to implement the Laplace Mechanism for satisfying ϵ-differential privacy. You will test your implementation on a set of integers randomly generated, where each integer represents the data of a distinct individual (*e.g.,* a salary).

1 Implementing Laplace

1. Launch Eclipse and create a Java project
2. Write a class `LaunchMe` that will serve as the entry point
   - Inputs :
     - The size n of the set of integers
     - The max value m of the integers (we fix the min value to 0)
     - The value of the privacy parameter ϵ
   - In `void main (String [])` :

— Generate the set of integers (you can use `double Math.random()` for generating a float uniformaly at random in `[0, 1[` and on `int Math.round(float)` in order to get the closest integer)

— Test on small sizes

3. Write a class called `Laplace` in charge of generating the Laplace perturbation such that :

— ϵ is a parameter of the `Laplace` object constructor

— The method `double Laplace.genNoise(int, double)` generates a random variable that follows the Laplace distribution and that is called before perturbing an aggregate query :

— The first parameter (type `int`) is the sensitivity of the agregate query to perturb

— The second parameter (type `double`) is a float in $]0, 1]$ representing the fraction of ϵ to consume for this call

— In order to generate a random variable that follows a Laplace distribution, you can (1) use `double Math.random()` for having a uniform random variable, and (2) apply the mathematical transform from a uniform variable to a Laplace variable described here : `https://en.wikipedia.org/wiki/Laplace_distribution#Generating_random_variables_according_to_the_Laplace_distribution`.

— Note that :

— A `Laplace` object stops returning any noise as soon as its ϵ is entirely consumed

— A `Laplace` object must have a `TEST` mode, which, when enabled, considers that the privacy budget is infinite (no budget consumption by any query)

2 Test your Implementation

4. **Test** `genNoise` as follows :

— Enable the `TEST` mode

— Choose an ϵ value and a sensitivity value

— Generate a large number of perturbations (*e.g.,* 10^4 should be enough)

— Count the number of perturbations that fall in the range $]-500, -480]$, those that fall in $]-480, -460]$, etc..., and those that fall in $]480, 500]$ (the histogram of the perturbations generated)

— Print the ranges and their corresponding counts in a CSV file, open the file with *e.g.* Open Office, and plot ranges and counts in a graph

— Compare your graph to the Laplace distribution that you should obtain (where b is sensitivity/ϵ) : `http://keisan.casio.com/exec/system/1180573177`

3 Analyze Laplace

5. **Why must we limit the number of aggregates published ?**
Enable the `TEST` mode. Let $n = 10^3$ and $\epsilon = 10^-4$:

— Formulate a `COUNT` on your set of integers (*e.g.,* count the number of integers greater than 10)

— Compute the true `COUNT` value : r

— Generate 10 perturbations from p_1 to p_{10}, obtain $r'_1 = r + p_1$ to $r'_{10} = r + p_{10}$ by perturbing r 10 times, and compute a_{10} the average of the r_i.

— Do the same with 10^2 perturbations, 10^3 perturbations, 10^4 perturbations, 10^5 perturbations, and 10^6 perturbations.

- Plot in a graph (*e.g.,* Open Office file) : on the x-axis the number of perturbations, and on the y-axis the averages. Plot also the true count
- How many perturbations are needed for being *close* to the true count (*e.g.,* $\pm 10\%$ difference) ?

6. **How big is the error due to the perturbation ?**
   Lets the *error* be the absolute value of the perturbation. Enable the `TEST` mode. Let $n = 10^3$, $\epsilon = 10^-2$, and $m = 1000$, :
   - Generate 10^3 perturbations for a `SUM` aggregate, compute the error due to each perturbation, and compute the average error err_{avg}
   - Does err_{avg} depend on the dataset size ? On the dataset values ?
   - What is the ratio between err_{avg} and the Laplace scale factor parameter (*i.e.,* sensitivity/ϵ) ?
   - With varying dataset sizes, *i.e.*, $n \in \{10^2, 10^4, 10^6\}$:
     (a) Compute the true `SUM` of the dataset values : *sum*
     (b) Compute the ratio between the previously computed average error and *sum* : err_{avg}/sum
     (c) Print n together with its corresponding ratio in a CSV file
   - Plot in a graph the evolution of the ratio with respect to the various dataset sizes n.
   - How many tuples are ≪ needed ≫ for making this ratio small enough ? (*e.g.,* 10%)