Imperial College
London

COMPUTATIONAL
PRIVACY
GROUP

Privacy Engineering

Week 4 - Formal privacy guarantees

Protecting a QBS against all possible attacks is hard

The system needs to:

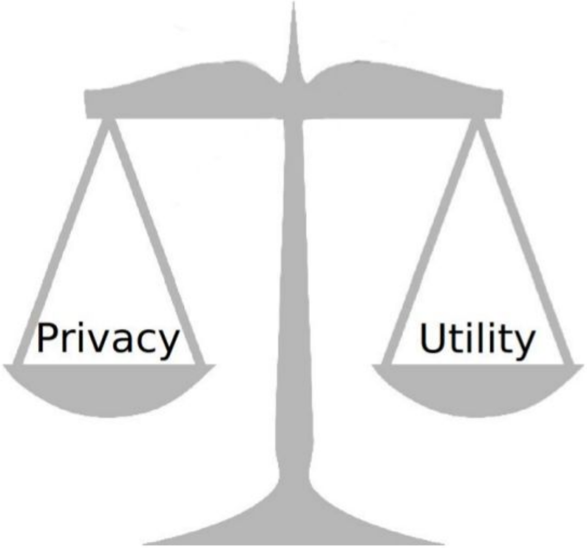
1) provide an expressive enough language/interface (to preserve utility)

2) yet be robust to all possible existing and future attacks

3) consistent (static and dynamic) noise addition and smart QSR help but as we have seen with Diffix is not sufficient (3 existing attacks, Aircloak's response so far has been to restrict the language)

However, we have so far always assumed that the attacker can run as many queries as he wants

How about limiting the number of queries?



Limiting the amount of information released

To limit the amount of information we’re releasing (and thereby prevent attacks), we could:

1. count the number of queries and stop answering when too many queries have been asked
2. increase the amount of noise we’re adding with each new queries.

But how can we know when to stop answering or how much noise to add? For this, we need to quantify how much information we are releasing with every answer

This is what **Differential Privacy** (which we’ll shorten to DP) has been developed for!

Differential Privacy: Idea

Differential privacy guarantees that the result of a certain query on the dataset must be essentially the same irrespective of the presence of any single individual in the dataset

$$\Pr[\text{output} = \text{y} \mid \text{Bob} \in D] \approx \Pr[\text{output} = \text{y} \mid \text{Bob} \notin D]$$

(for any possible output y and for any arbitrary user)

Here the probability is computed with respect to the **randomization of the algorithm**. It was just the same with simple noise addition: we had a mechanism that computes something (e.g. a count) and then adds some **random** noise according to some probability distribution (e.g. Gaussian).

Why could this be a valid privacy definition?

To get perfect privacy, we'd however like to have "=" instead of "≈". Why would this completely destroy utility?

Differential Privacy: Intuition

Instead, we bound the difference between the dataset with and without Bob (or any other user) using ε:

$$1 - \varepsilon \lesssim \frac{\Pr[\text{output} = \text{y} \mid \text{Bob} \in D]}{\Pr[\text{output} = \text{y} \mid \text{Bob} \notin D]} \lesssim 1 + \varepsilon$$

If ε is small, 1 − ε and 1 + ε will both be close to 1 and the output with and without Bob (or any other user) will be very similar

More formally, we use the notion of *neighboring dataset* to refer to the datasets with (D1) and without Bob or any other users (D2).

Def: We define a dataset as a collection of rows, where each row is the record of a different user. We will use $\mathbb{D}$ to refer to the collection of all datasets. We say that two different datasets D1 and D2 are **neighboring** if they differ by exactly one row, which means D1 = D2 ∪ {r} or D2 = D1 ∪ {r} (where r is some row).

A mechanism is ε-Differentially private if it satisfies this definition.

Differential Privacy: Formal definition

A randomized algorithm M: D → ℝ is ε-differentially private (ε-dp) if for any two neighboring datasets D and D' and for every possible output y we have:

$$\Pr[\mathcal{M}(D) = y] \leq e^{\varepsilon} \Pr[\mathcal{M}(D') = y]$$

Remarks:

- M is the algorithm applied to the dataset to both compute the output and ensure privacy. Here M takes values in ℝ for simplicity, but could be any set.
- The probability space is over the randomization (or “coin flips”) of M
- By symmetry we can swap D and D' in the equation. Combining the two equations we get $e^{-\varepsilon} \leq \frac{\Pr[\mathcal{M}(D) = y]}{\Pr[\mathcal{M}(D') = y]} \leq e^{\varepsilon}$
- Since $e^{-\varepsilon} \approx 1 - \varepsilon$ and $e^{\varepsilon} \approx 1 + \varepsilon$ for $\varepsilon \approx 0$, we obtain $1 - \varepsilon \lesssim \frac{\Pr[\mathcal{M}(D) = y]}{\Pr[\mathcal{M}(D') = y]} \lesssim 1 + \varepsilon$
- And if $\varepsilon \approx 0$ we get $\Pr[\mathcal{M}(D) = y] \approx \Pr[\mathcal{M}(D') = y]$

Differential Privacy explained

The definition ensures that the output of M will be essentially (as quantified by ε) the same, independent of whether any individual is in the dataset or not.

This is how Differential Privacy defines privacy. It is an “arbitrary” choice but one that gives strong meaningful guarantees and works well mathematically.

Note: like for QBS, Differential Privacy considers everything sensitive and makes no distinction between quasi-identifiers and sensitive information. Everything has to be protected incl. belonging to the dataset

Note: The original definition of DP is user-centric (as in the definition focuses on hiding the presence or absence of a user) but can be adapted to other contexts

What DP doesn’t protect

Remember our results of illegal drug testing example?

I have a dataset D with the drug testing results of people all over the UK. Using a differentially private mechanism, I compute the percentages of people in the dataset that test negative to illegal drugs and the percentage of people in South Kensington that test negative to illegal drugs.

The results are: 99.1% overall and 15.5% in South Kensington.

If I know you’re in the dataset and live in South Kensington did I learn something about the likelihood of you using illegal drugs?

Achieving DP: Laplace mechanism

We’ve seen before that adding (consistent) noise helps protect privacy.

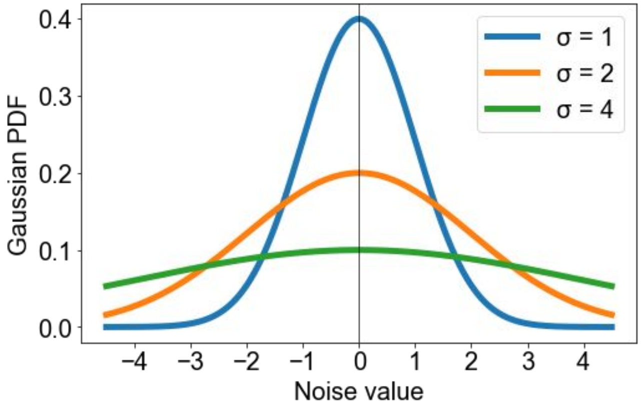
Noise addition is often one of the key ingredients to designing a DP mechanism.

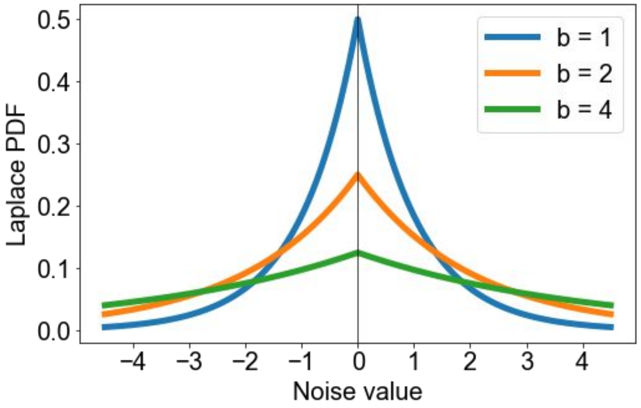
For DP, we however generally use Laplace noise instead of the Gaussian noise we used before.

The pdf of a Laplace distribution is:

$$f(x \mid \mu, b) = \frac{1}{2b} \exp\left(-\frac{|x-\mu|}{b} \right)$$

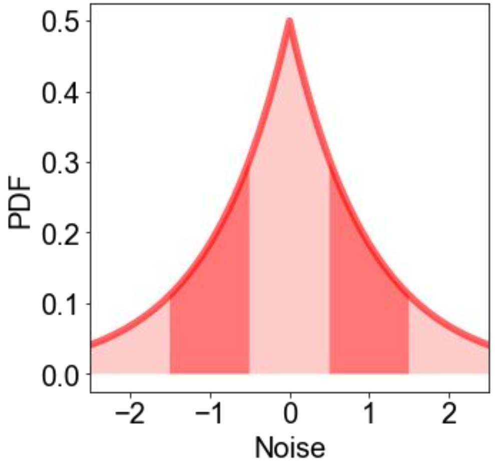
which has mean μ and variance $2b^2$. We denote by Lap(b) the distribution with pdf *f(x | 0, b)*.

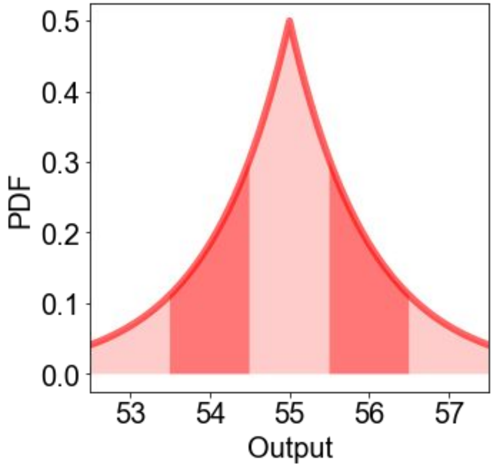




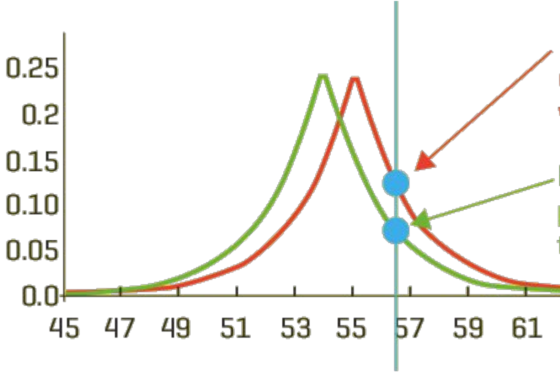
Achieving DP: Laplace mechanism

Going back to our query from last week: “How many students born in 1995 code with Notepad?”. Suppose that we add Laplace noise drawn from Lap(1/ε) to the true value. For example, if the true value is Q(D) = 55, we’ll answer 55 with a probability of 0.39, 54 with a probability of 0.19, etc.





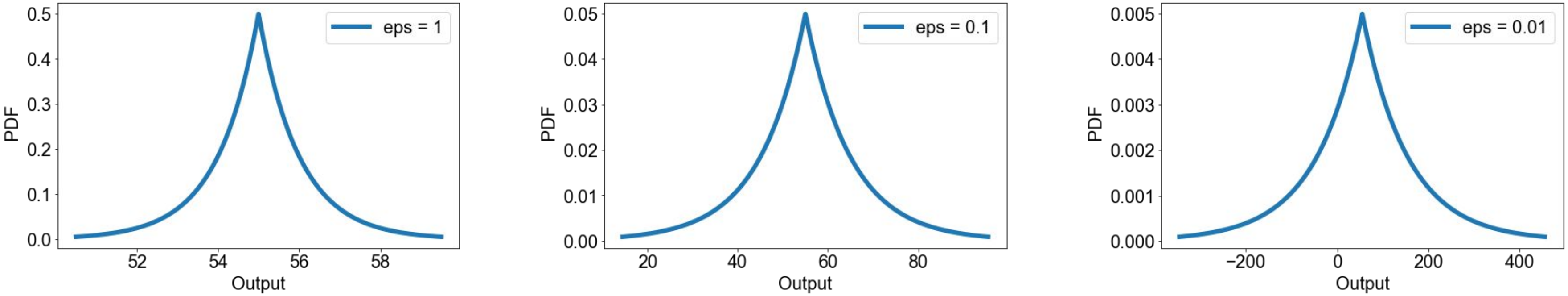
One can then prove that (for one query) this mechanism M that returns Q(D) + Lap(1/ε) is ε-differentially private.



Probability that the privatized count came from the possible world where Bob took the survey : 62%

Probability that it came from the possible world where Bob didn’t take the survey: 38%

The smaller the ε, the more noise we add



Note: in DP parlance, people often say that we add *noise proportional to 1/ε.* This is an **abuse of language** to mean that we are adding noise using a Laplace distribution which has a standard deviation σ = √2/ε

This, however, can be confusing as it **does not** mean that we are adding noise as a function of the output we're trying to protect (55). In general this is risky, and the Laplace mechanism relies on data independent noise. Using data dependent noise with DP is possible, but requires much more sophisticated mechanisms.

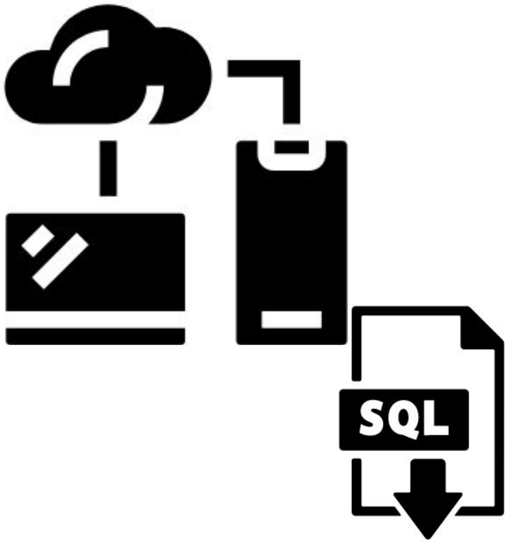
DP and attacks (one query)

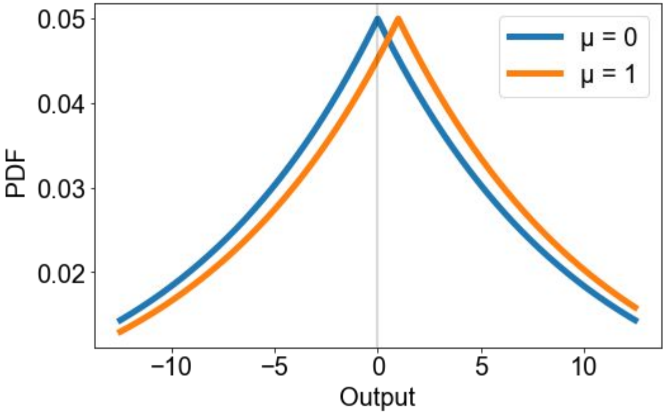
Going back to our exemple, we now have a server with the same dataset of DoC students which we protect using Differential Privacy using an ε = 0.1 and the Laplace mechanism (thereby adding noise *proportional to 1/0.1* = 10)

We now ask our server: “How many students are born on 1994-09-23 and code with Notepad?”

Even if we know that there is one and only one person in the dataset born in 1994-09-23, with one question, our guess will be barely better than random

Naturally, this also thwarts intersection attacks without the need for QSR



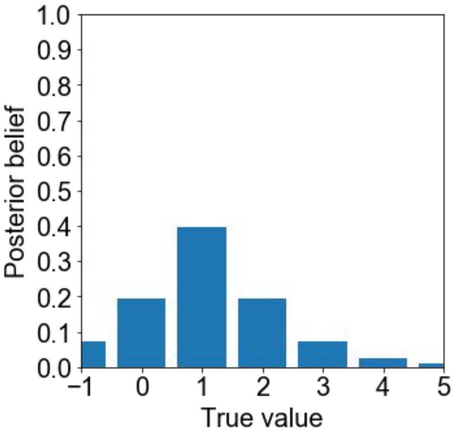


Ok, but how about averaging attacks?

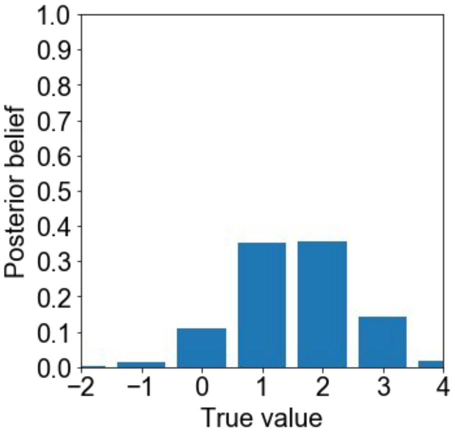
Well… they work

This is expected, we’re adding (potentially, a lot of) noise but the CLT still applies

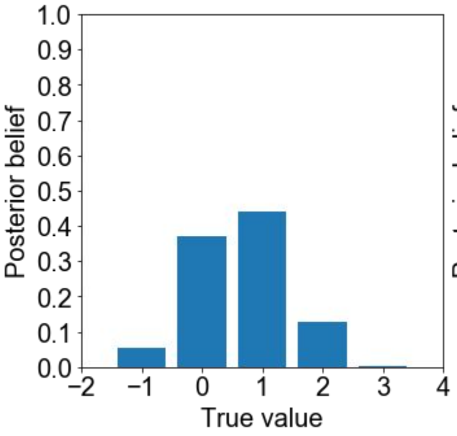
Running our (Bayesian) averaging attack, we need 10 queries to learn with high likelihood that Bob codes in Notepad. Here is the posterior when ε=1.



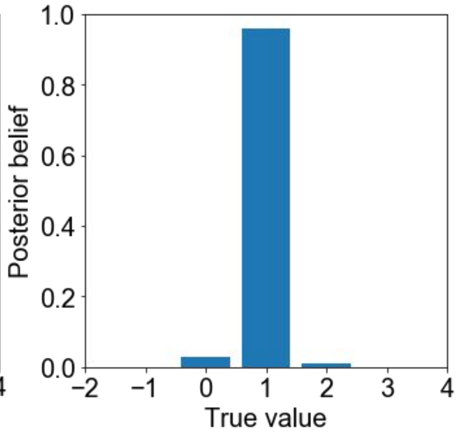
1 query



2 queries



5 queries

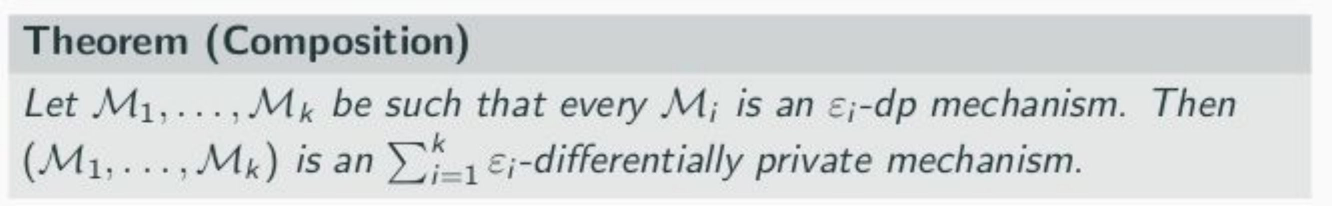


10 queries

Calibrating the noise to the number of queries

And this is where the key idea of differential privacy comes in: decide how many queries **will ever be ran on this dataset** and ensure that we're adding noise proportionally. In our previous example, adding noise proportional to ε=1 was sufficient to protect Bob's (or anyone in the dataset's) privacy against 5 queries but not against 10 of them.

This is where the nicest feature from DP comes in: composability. Releasing the output of **any** two queries on a dataset protected by a ε-DP mechanism is equivalent to releasing one query protected by a 2ε-DP mechanism



Theorem (Composition)

Let $\mathcal{M}_1, \ldots, \mathcal{M}_k$ be such that every $\mathcal{M}_i$ is an ε_i-dp mechanism. Then $(\mathcal{M}_1, \ldots, \mathcal{M}_k)$ is an $\sum_{i=1}^{k} \varepsilon_i$-differentially private mechanism.

We can then decide on an ε, which we call the privacy budget, which then defines the total number of queries (any of them!) anyone can run on the dataset

Privacy budget: Example

One can see the DP privacy budget idea as a **worst case analysis**, every query has a cost and we will add them up such that, even if every query you asked is “How many students are born on 1994-09-23 and code with Notepad?”, your guess on whether Bob codes with Notepad will not increase significantly **because Bob was in the dataset**. In other terms, if your guess about Bob changes, then it would have changed almost in the same way if Bob didn’t take part in the survey.

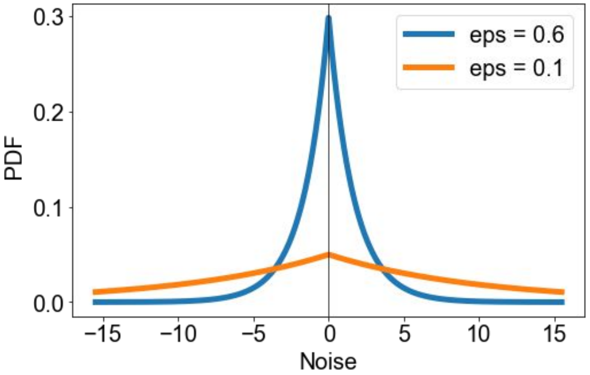
Formally, one considers two posterior beliefs corresponding to two different “worlds”: the one where Bob is in the dataset (post_{Bob}) and the one where he’s not ($\text{post}_{\text{-Bob}}$). If ε-DP is enforced, this guarantees that $\text{post}_{\text{Bob}}(x) \leq e^{\varepsilon}\, \text{post}_{\text{-Bob}}(x)$ for any guess x. That is, the **two posteriors** are very close, **whatever the prior**.

In practice, if $\varepsilon=1$ and $\text{post}_{\text{Bob}}(0) = 0.5$, then $0.18 \leq \text{post}_{\text{-Bob}}(0) \leq 0.82$

Using the budget

Actually, the composability theorem allows us to go even further: as long as we account for the cost of every query we can spend it any way we want!

For instance, we could decide to spend 0.60 on the first query (what we really want to know), e.g. the number of student in DoC born in 1994 who use Notepad and then 0.1 on 1992, 1993, 1995, and 1996.



We'll add noise of Lap(0.6) to the first query and then Lap(0.1) on the other four queries

The effect of the caching or seeded PRNG can also be taken into account in the DP framework: as they're not releasing any extra information they're free ε=0

Can you give every user his or her own privacy budget?

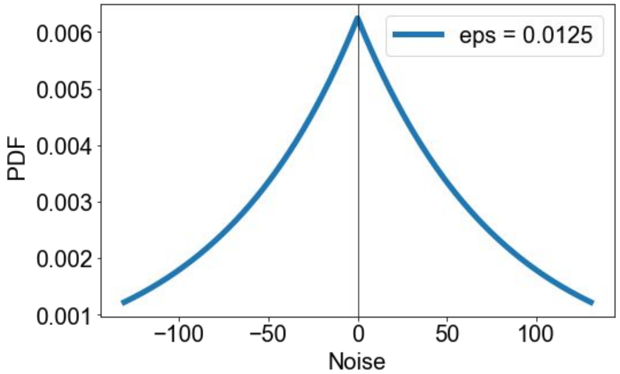
Is this an issue?

DP and utility

While extremely nice and formal, DP relies on a worst-case analysis. Either it adds a lot of noise or it only allows you (and everyone else) to ask a small number of questions before you have to destroy the dataset

For instance, assume you'd want to look at the editor used by male and female students in DoC by age group

You'd need to ask for each editor (4: Notepad, Vim, Emacs, Atom), for both gender (2) and then for each age group (10: say 1990 to 2000). That'll be 80 queries right there. Assuming these are the only ones you'll ever want to ask you'd already have to add Lap(1/80) of noise to every answer to give individual the $e^1 \approx 2.7$ guarantee we had before.



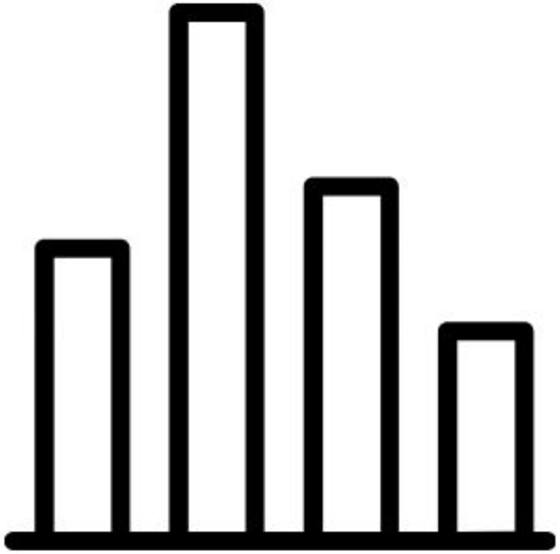
Optimizing for DP: Histograms

A large body of research has (and still is) developed methods to **decrease the cost of specific queries**

One type of such queries are **histograms**

For instance, in our dataset, an analyst might want to compute the fraction of people in DoC who use Notepad, Sublime, Atom, Vim, and Emacs

Other are mechanisms that release evolving data in real time (e.g. the population density of a region), or mechanisms for ML and even DL.

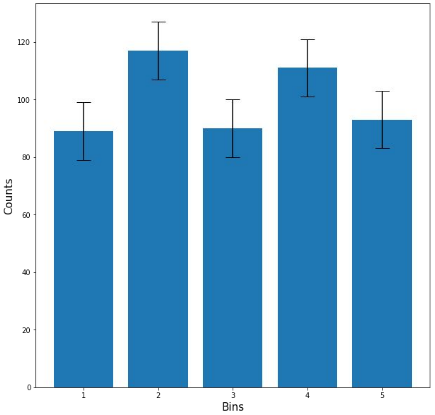


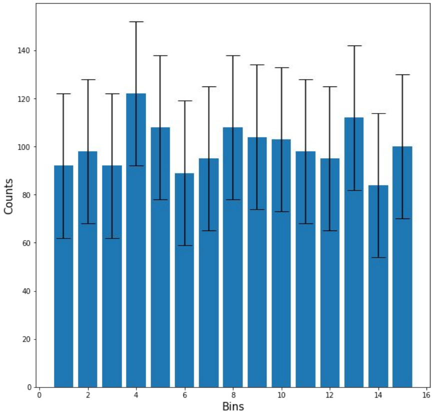
Histograms the non-optimized way

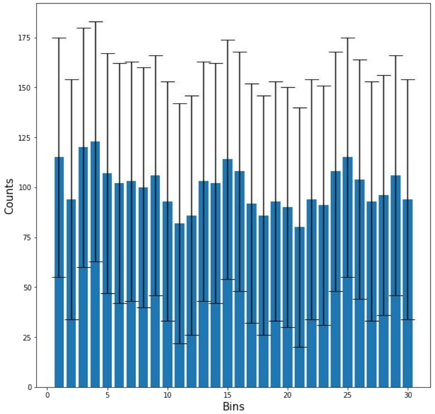
The naive way would be to ask: Q_X = “How many users code with text editor X” for every X: Notepad, Sublime, Atom, Vim, Emacs, adding noise to every query

We’d however have to split the privacy budget among all the queries Q_X as seen before. With a privacy budget of ε = 1, we would have to release every query Q_X with privacy budget ε/X = 1/5 = 0.2 resulting in Lap(0.2) which is quite reasonable

However, if we had more text editors (e.g. because we’re looking at the combination of year and month of birth with gender), this would quickly become prohibitive. We’d have to add 1/(48*2), Lap(960) of noise!

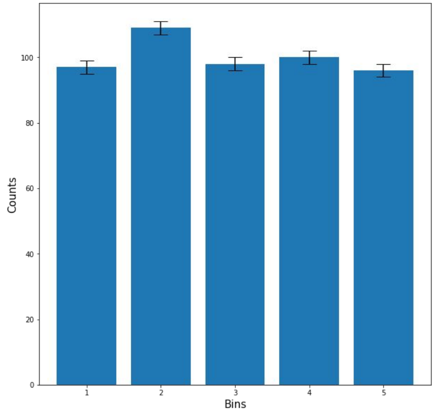






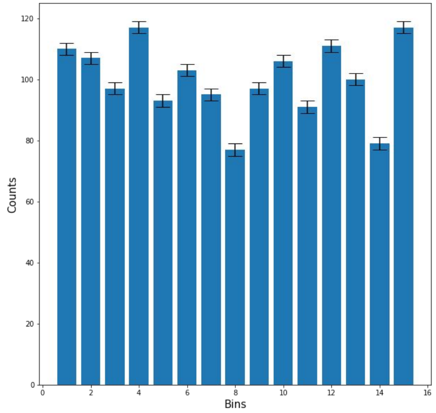
© CO408 - Yves-Alexandre de Montjoye

Histograms the optimized way

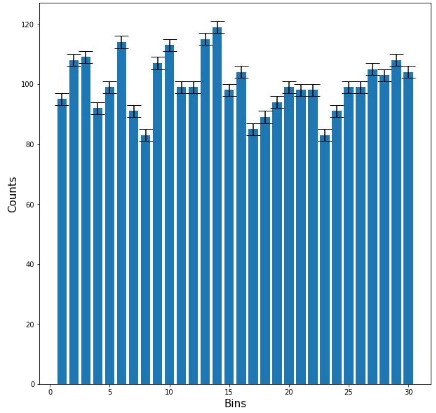


Fortunately, there is a much better solution.

Indeed, every user in the dataset contributes to only one bucket in the histogram. This means that removing or adding a single user from the dataset can affect the count of only one bucket.



It’s easy to prove that this allows us to not have to split the privacy budget. We can release the histogram by adding Laplace noise with parameter 1/ε to each bucket count. The noise doesn’t scale anymore with the number of buckets making it possible.



In fact, we can prove the same result for any sequence of queries (not only histograms) where each query is about one and only one (vertically) disjoint subset of the dataset.

What if I want to do more than counting?

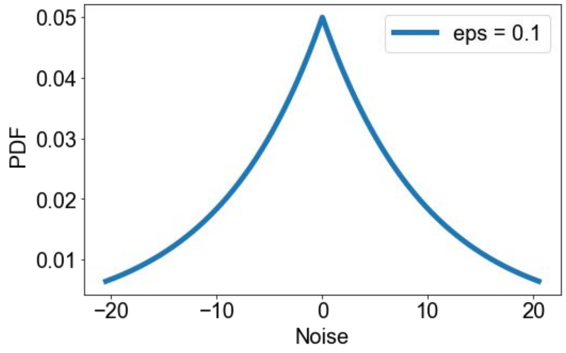
E.g. my dataset could have a scholarship column and I could ask the question: “What is the sum of all scholarships of students born on 1994-09-23?”

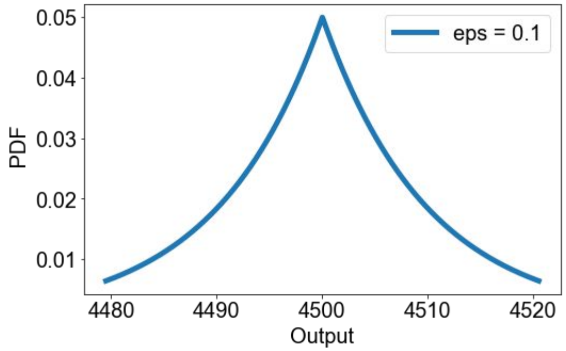
Would adding noise according to a Lap(10) be sufficient?

That’ll certainly not be sufficient to prevent us from learning with high accuracy the amount of Bob’s scholarship. Indeed, 63% of the time the true value of the scholarship will be within the answer +/- 10.

Here are 5 samples from 4500 + Lap(10):

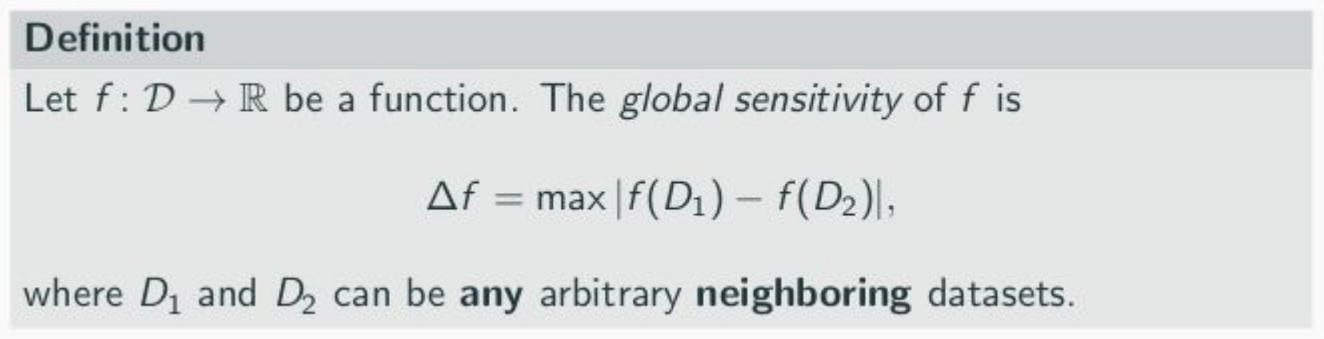
[4494, 4504, 4496, 4507, 4489]





Protecting general queries: Function Sensitivity

To protect privacy here, we need to add noise proportional to the maximum scholarship any individual could receive: The global sensitivity of a function f captures the magnitude by which a single individual’s data can change the function *f* in the worst case, and therefore, the uncertainty in the response that we must introduce in order to hide anyone’s participation.



Definition

Let $f : \mathcal{D} \rightarrow \mathbb{R}$ be a function. The *global sensitivity* of f is

$$\Delta f = \max |f(D_1) - f(D_2)|,$$

where D_1 and D_2 can be **any** arbitrary **neighboring** datasets.

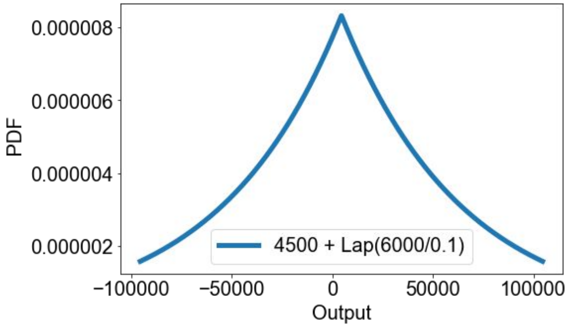
In our example, *f* is the sum of all scholarship of students born on 1994-09-23. Since the maximum annual scholarship at IC is £6000, we have that ∆f = 6000.

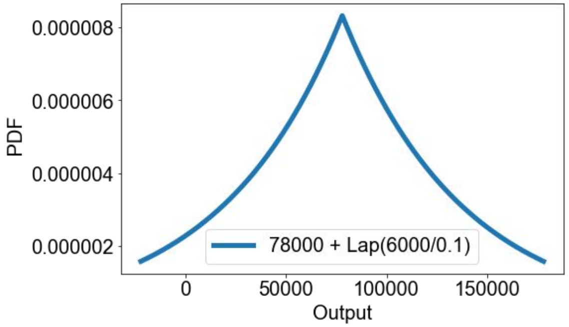
Function Sensitivity

Adding noise according to Lap(∆f/ε)

- We protect individuals. Answers to the query “What is the sum of all scholarships of students born on 1994-09-23?” do not tell us much about Bob’s scholarship amount at all or even if he got one.

- Yet allow an analyst to ask legitimate questions such as: “What is the total sum of scholarships at DoC?” and others questions for which the amount of noise added by a Lap(60000) is more “reasonable”

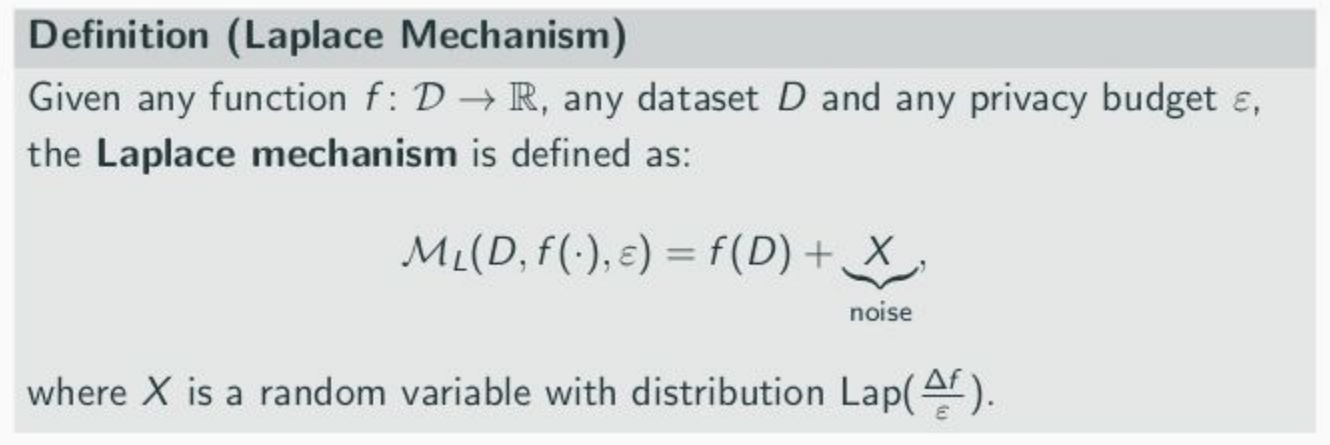




Function sensitivity and privacy budget

Will this be sufficient to prevent averaging attacks?

Actually yes, this is why we introduced a second parameter (global sensitivity) on top of ε. Using the function sensitivity, we can rest assured (theorem) that adding noise proportional to Δf/ε will give us an ε-differentially private mechanism



Definition (Laplace Mechanism)

Given any function $f\colon \mathcal{D} \to \mathbb{R}$, any dataset D and any privacy budget ε, the **Laplace mechanism** is defined as:

$$\mathcal{M}_L(D, f(\cdot), \varepsilon) = f(D) + \underbrace{X}_{\text{noise}},$$

where X is a random variable with distribution $\text{Lap}(\frac{\Delta f}{\varepsilon})$.

Group privacy

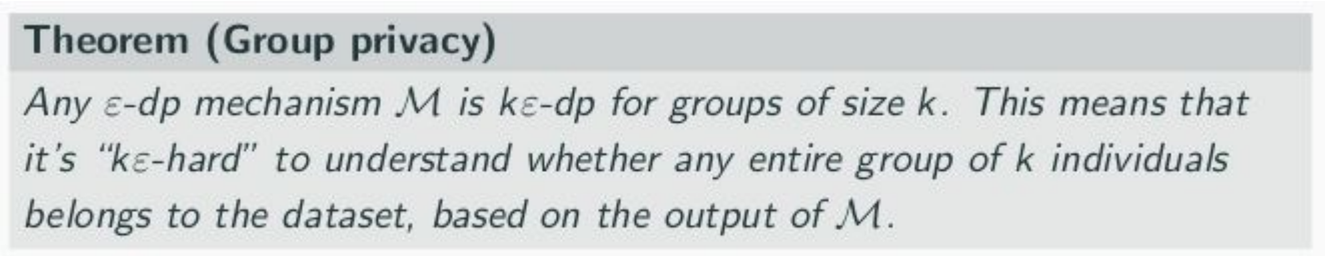
Last week, we looked at whether ad-hoc privacy-preserving mechanisms and, more specifically, bounded noise addition were protecting from (“powerful”) attackers interested in finding out information about groups.

We had assumed that the attacker knew that:

- There are 10 male DoC students born in April 1994
- They all use the same text editor

Would Differential Privacy protect us against this?

Kind of...



Theorem (Group privacy)

Any ε-dp mechanism $\mathcal{M}$ is $k\varepsilon$-dp for groups of size k. This means that it's “$k\varepsilon$-hard” to understand whether any entire group of k individuals belongs to the dataset, based on the output of $\mathcal{M}$.

In short, DP guarantees can be extended to groups of size k but at the cost of “multiplying the noise” by a factor k. So to protect a group of 10 people we’d have to go from Lap(10) to Lap(100)

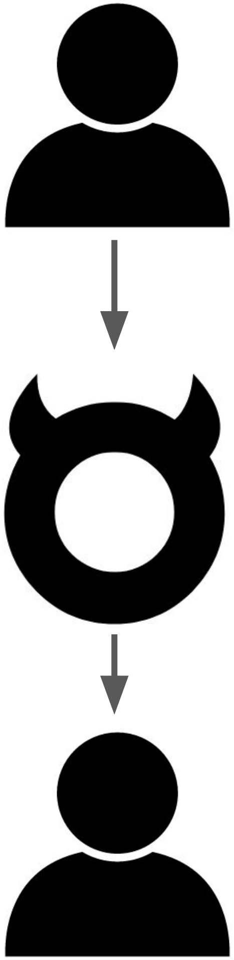
What if we don’t trust a third party?

But what if we don’t have a QBS system? Or even trust someone to ever collect all the data?

So far in this course, we have always worked in the following setup with three entities:

- the users, who provide the data;
- the trusted curator, who collects the data and acts as mediator;
- the data analyst, who receives differentially private answers from the

There are two tools that address this problem: multiparty computation (MPC) and local differential privacy



Two approaches: MPC and local DP

Multiparty computation:

In MPC, the main idea is have the users perform the computation together, in a distributed fashion. MPC guarantees cryptographically that no user can learn the data of anybody else (under certain general hypothesis) all while allowing the analyst to receive the output.

Local differential privacy:

In local differential privacy, every user “adds noise” to his own data (=locally) and then shares the “privatized” data with the analyst. The analyst can then run any computation on these privatized data.

Both approaches address partially different privacy and utility problems and researchers are now exploring ways to combining them.

In this lecture, we will now focus on local DP.

Local DP: Randomized response

The idea of standard DP is that the output on a dataset should not reveal whether a user belongs to the dataset. In local DP, participation is not a secret anymore. **The only secret is the individual response**.

Example: Suppose a professor wants to conduct a survey among students asking the question: “Have you cheated at the exam?”

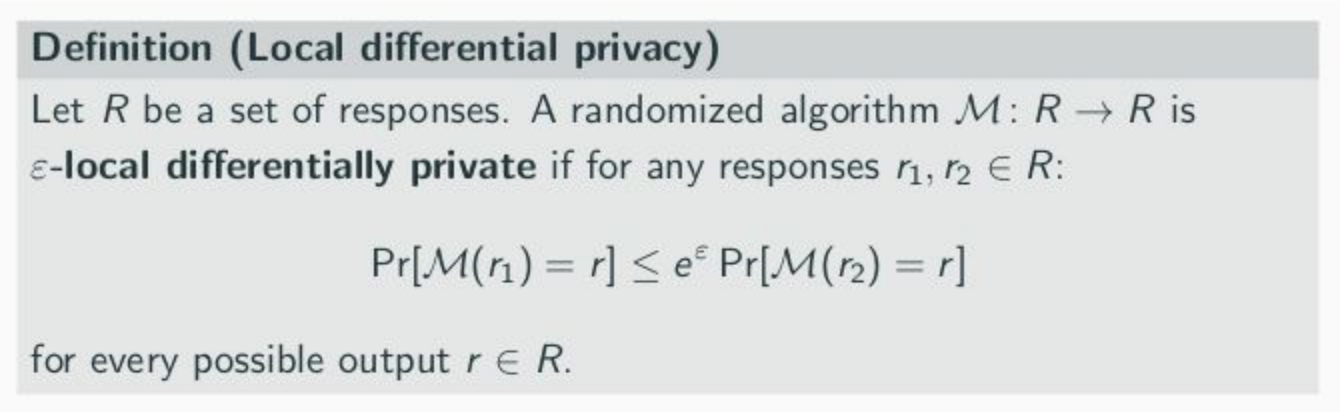
Every student is asked to answer YES or NO, following these steps:

1. Flip a biased coin, with probability of tails $p > 0.5$
2. If tails, then respond truthfully
3. If heads, then lie.

This procedure is called **randomized response**

Local DP: Definition

The definition (and therefore guarantees) of Local Differential Privacy are different from the ones of Differential Privacy. Local DP however enjoys similar properties



Definition (Local differential privacy)

Let R be a set of responses. A randomized algorithm $\mathcal{M}: R \rightarrow R$ is ε-**local differentially private** if for any responses $r_1, r_2 \in R$:

$$\Pr[\mathcal{M}(r_1) = r] \leq e^{\varepsilon} \Pr[\mathcal{M}(r_2) = r]$$

for every possible output $r \in R$.

The randomized response is an example of a local-dp algorithm, where R = {YES, NO}. Given p(>0.5), what is the ε?

It’s easy to check (exercise) that $\varepsilon = \log(p/1-p)$

Local DP: Real-world implementations

Most of not all current real-world systems using Differential Privacy implement local DP

Google released RAPPOR in 2014. RAPPOR is a toolkit of local DP mechanism that Google uses to collect “anonymous” data about Chrome usage to detect unexpected behavior (e.g. malware).

Apple developed its implementation of local DP in 2017. Apple collects data about websites’ usage of resources on Safari and about words typed by users to improve predictions.

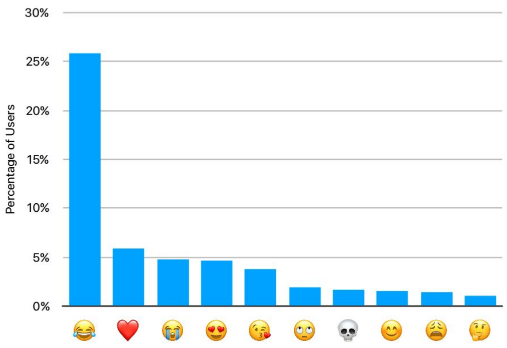
Local DP however tends to add a lot of noise, and state-of-the-art mechanisms require to compromise on privacy guarantees in order to attain good utility. For example, Apple’s implementation of DP uses values of ε as high as 8, the equivalent of lying less than 0.03% of the times in a binary case (although other heuristics are in place to significantly reduce the risk of data leakage).

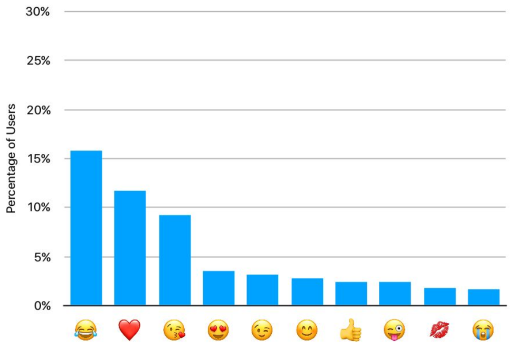
Challenges in using DP: Local DP and evolving data

When using local DP, very often the data collector wants to collect data many times e.g. to study how it evolves over time

In the case of Apple, they collect usage statistics from iOS roughly every day, each time resetting the ε parameter. This means that their DP mechanism ignores **any potential correlations over time.**

From a theoretical point of view at least, this is a major weakness in Apple's implementation.





Challenges in using DP: Graphs

The standard (not local) definition of DP strongly depends on the choice of neighboring datasets. So far, we have always worked with datasets that were a table with every row corresponding to one (and only one) user. Our definition of neighboring datasets (removing/adding a row) therefore meant protecting a user.

There are cases where defining neighboring datasets is not so easy.

For example, consider the case of a social network, where the nodes are users and edges are friendship relations. How do we define the neighbor of a graph?

The strongest definition is to, again, consider that we want to hide any user: G1 and G2 are neighboring if G1 can be obtained from G2 by removing a single node and all of its incident edges, or vice versa.

This, however, requires adding a lot of noise for many types of queries. For example, consider the query that releases the total number of edges. This function has unbounded sensitivity, as there is no limit on the number of friends!

Summary:

Week 0: Intro to privacy

Week 1: Data pseudonymization and anonymization

Secret formulas, hash functions and lookup tables, k-anonymity, etc

Week 2: Anonymization revisited, privacy of big data

Unicity, data generalization, matching and profiling attacks, privacy of unstructured data, etc

Week 3: Query-based systems

Privacy of query-based systems (e.g. SQL), averaging and intersection attacks, etc

Week 4: Formal guarantee for privacy

Differential privacy (2017 Gödel Prize), Laplace mechanism, group privacy, etc

Privacy is an (hard but highly enjoyable) attack and defense game. There is not perfect solution (almost by definition), there will always be trade-offs but many problems we can strongly limit the risks if we care and use the right technical tool for the task!

Questions?