Privacy for Computations

Vicenç Torra

February 2024

Umeå University, Sweden

V. Torra (2022) A guide to data privacy, Springer (Chapter 5)

Outline

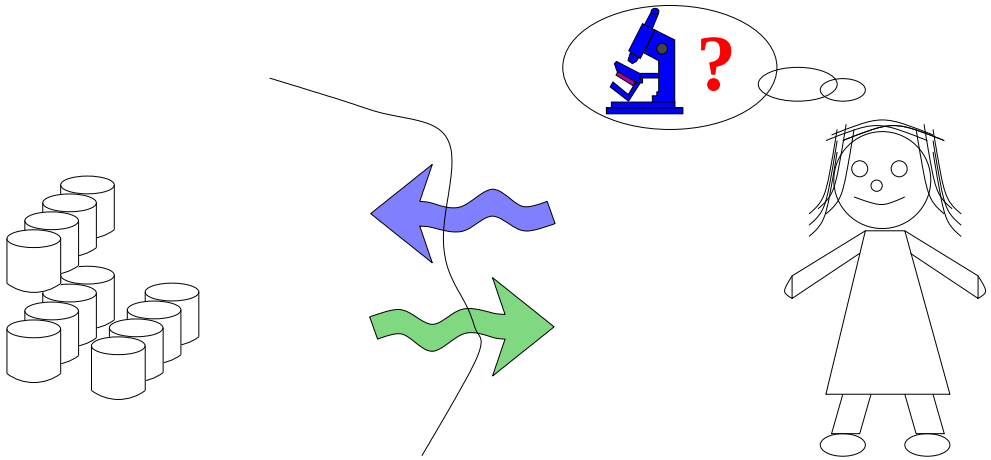
1. Computation-driven approaches
   - Differential privacy
   - Centralized approach: trusted third party
   - Distributed approach: secure multiparty computation

Privacy for computations

Introduction

Introduction

- The researcher computes a function without accessing the data

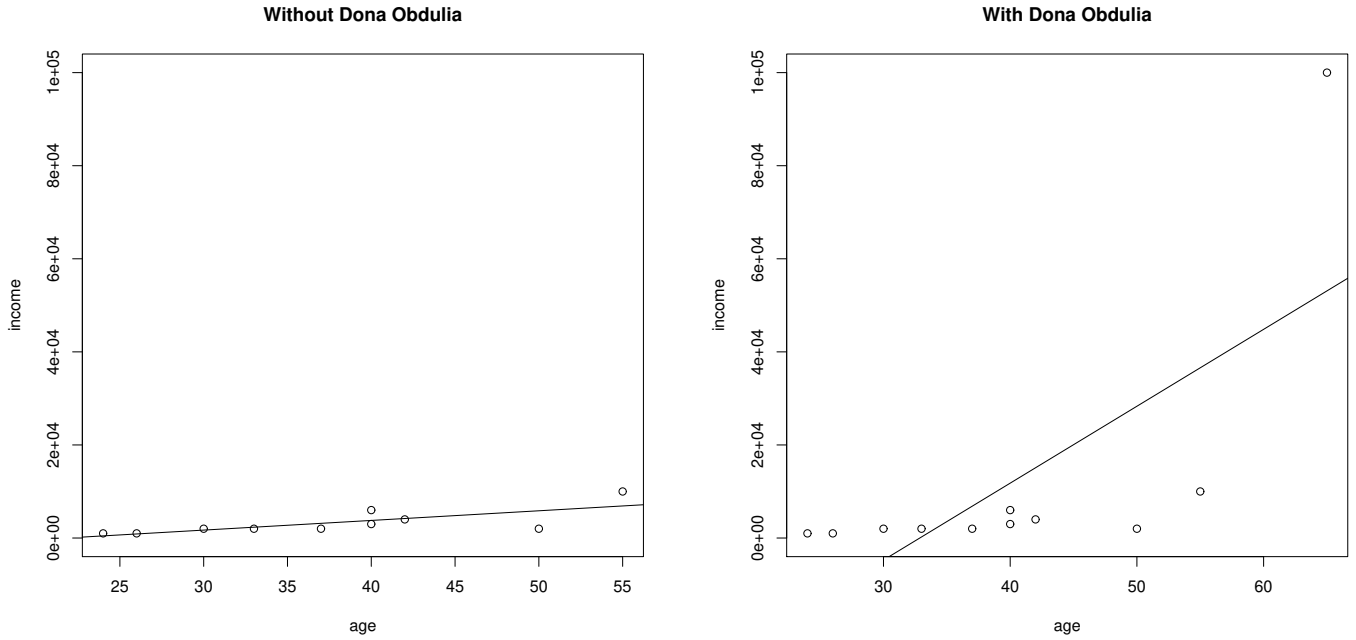


Data is sensitive: computation leads to disclosure

- Motivating example #1 (Case #2. Sharing a computation)
  - Q: Mean income of admitted to hospital unit (e.g., psychiatric unit) for a given Town (Bunyola)?
  - Mean income is not "personal data", is this ok ? NO!!:
  - Example 1000 2000 3000 2000 1000 6000 2000 10000 2000 4000 $\Rightarrow$ mean = 3300
  - Adding Ms. Rich's salary 100,000 Eur/month: mean = 12090,90 !
    (a extremely high salary changes the mean significantly)
    $\Rightarrow$ We infer Ms. Rich from Town was attending the unit

Data is sensitive: computation leads to disclosure

- Motivating example #2 (Case #2. Sharing a computation)



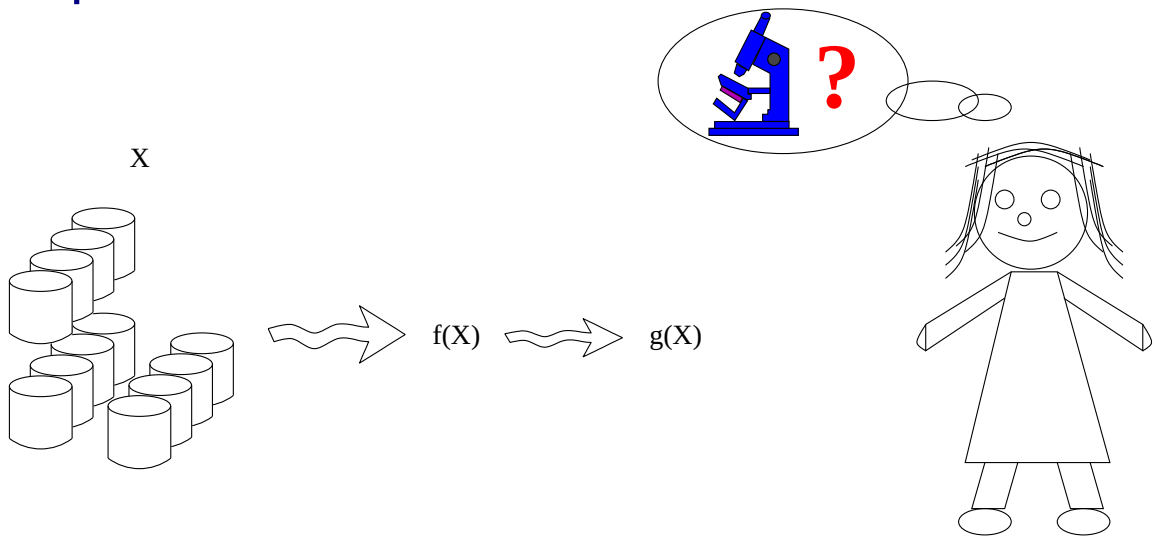
- Regression of income with respect to age with (right) and without (left) the record of Dona Obdúlia
  - income = -4524.2 + 207.5 age (without Ms. Rich = Dona Obdúlia)
  - income = -54307 + 1652 age (with Ms. Rich = Dona Obdúlia)

Privacy models (review)

Privacy for computations: privacy models I

Privacy models. Computing a function (centralized)

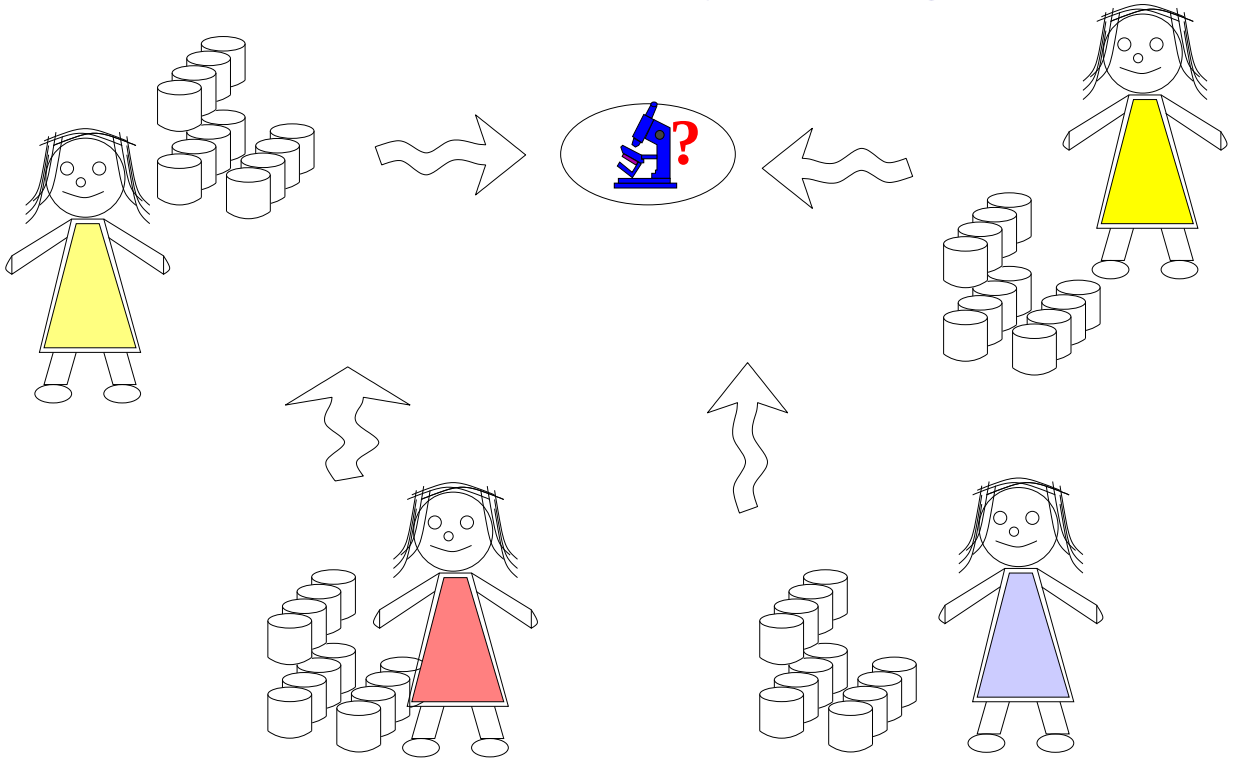
- **Differential privacy.** The output of a query to a database should not depend (much) on whether a record is in the database or not.
- **Integral privacy.** Inference on the databases. E.g., changes have been applied to a database.
- **Homomorphic encryption.** We want to avoid access to raw data and partial computations.



Privacy for computations: privacy models II

Privacy models. Computing a function (distributed)

- **Secure multiparty computation.** Several parties want to compute a function of their databases, but only sharing the result.



Introduction: Summary

- Important assumptions
  - We know the function to compute
  - Data is not shared, only the output of the function
  - Partial computations are not shared, only the output of the function
  - We do not want that the output of the function leads to disclosure

Introduction: Summary

Differential privacy

Privacy model: definition

Differential privacy

- Computation-driven/single database
  - Privacy model: differential privacy[1]
  - We know the function/query to apply to the database: f
- Example:

  compute the mean of the attribute salary of the database for all those living in Town.

[1] There are other models as e.g. query auditing (determining if answering a query can lead to a privacy breach), and integral privacy

Differential privacy

- Differential privacy (Dwork, 2006).
  - Motivation:
    - ▷ the result of a query should not depend on the presence (or absence) of a particular individual
    - ▷ the impact of any individual in the output of the query is limited

  differential privacy ensures that the removal or addition of a single database item does not (substantially) affect the outcome of any analysis (Dwork, 2006)

Differential privacy

- Mathematical definition of differential privacy
  (in terms of a probability distribution on the range of the function/query)
  - A function K_q for a query q gives ϵ-differential privacy if for all data sets D_1 and D_2 differing in at most one element, and all $S \subseteq Range(K_q)$,

$$\frac{Pr[K_q(D_1) \in S]}{Pr[K_q(D_2) \in S]} \leq e^{\epsilon}.$$

    (with 0/0=1) or, equivalently,

$$Pr[K_q(D_1) \in S] \leq e^{\epsilon} Pr[K_q(D_2) \in S].$$

- ϵ is the level of privacy required (privacy budget). The smaller the ϵ, the greater the privacy we have.

Differential privacy

- Differential privacy: A KEY ELEMENT
  - for all data sets D_1 and D_2
    differing in at most one element

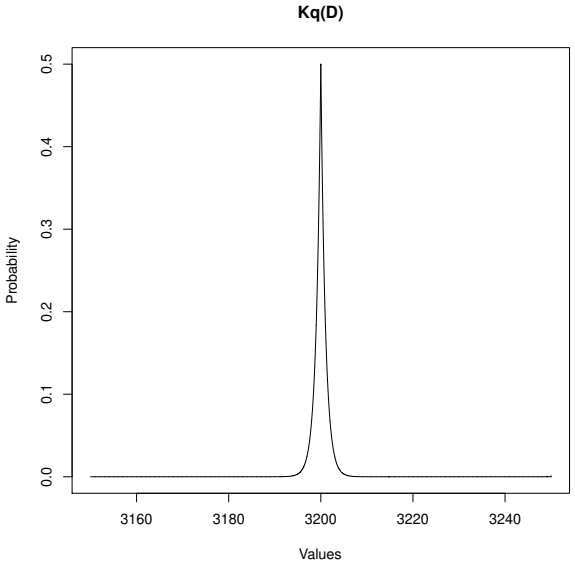
Understanding the definition:

Differential privacy for numerical data

Differential privacy

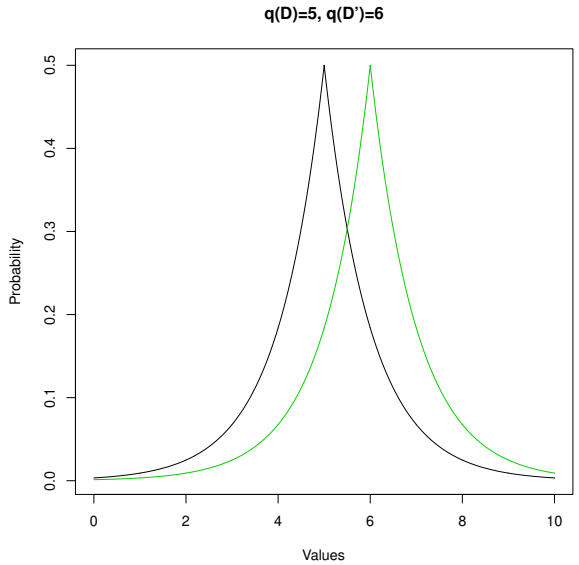
- Differential privacy
  - A function K_q for a query q gives ϵ-differential privacy if . . .
    - $K_q(D)$ ~~is a constant. E.g.,~~ $~~K_q(D) = 0~~$
    - $K_q(D)$ is a randomized version of $q(D)$:

$$K_q(D) = q(D) + and\ some\ appropriate\ noise$$



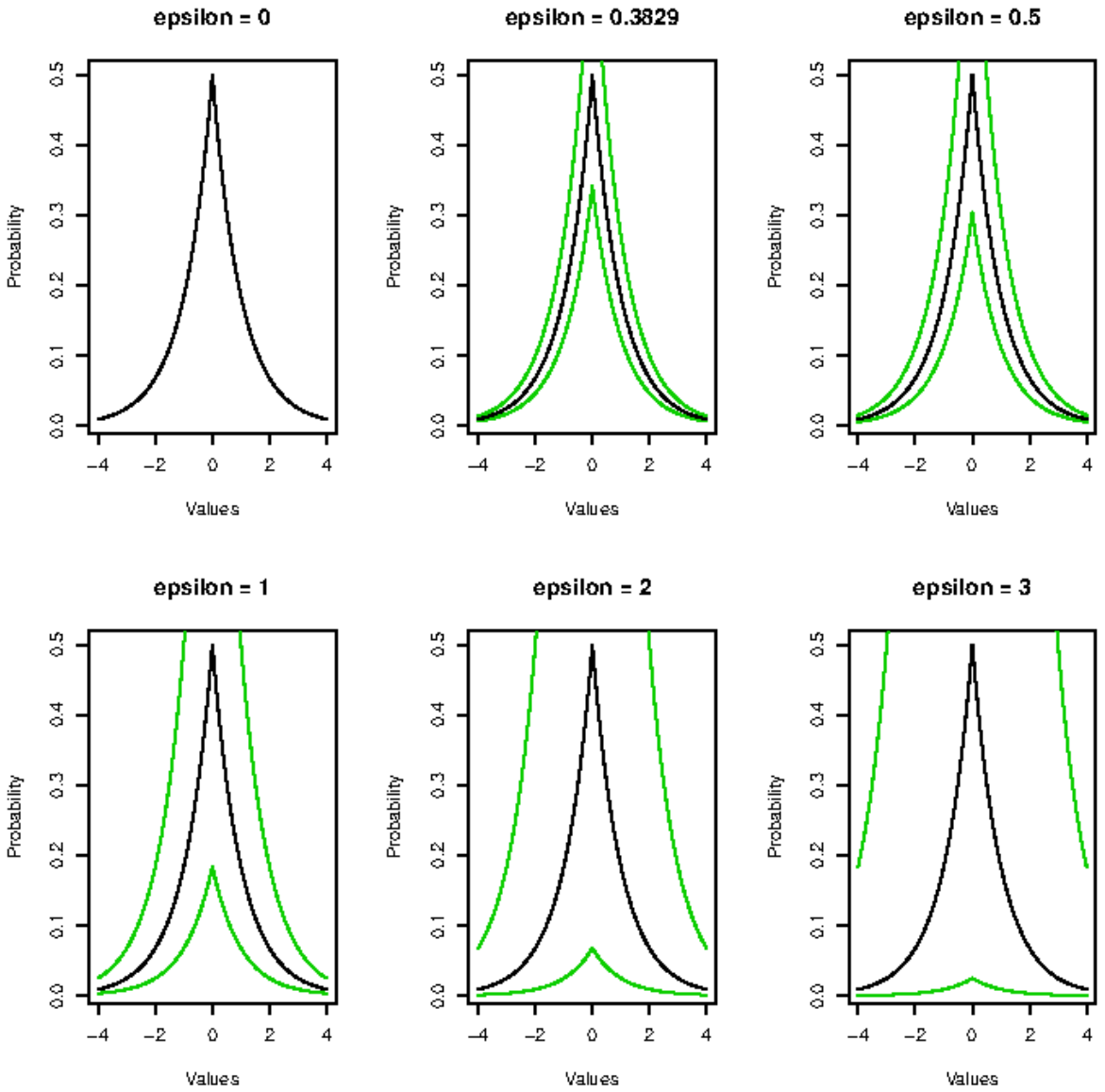
Differential privacy

- Differential privacy
  - $K_q(D)$ for a query q is a randomized version of $q(D)$
    - ▷ Given two neighbouring databases D and D'
      $K_q(D)$ and $K_q(D')$ should be similar enough . . .
  - Example with $q(D) = 5$ and $q(D') = 6$ and adding a Laplacian noise $L(0, 1)$



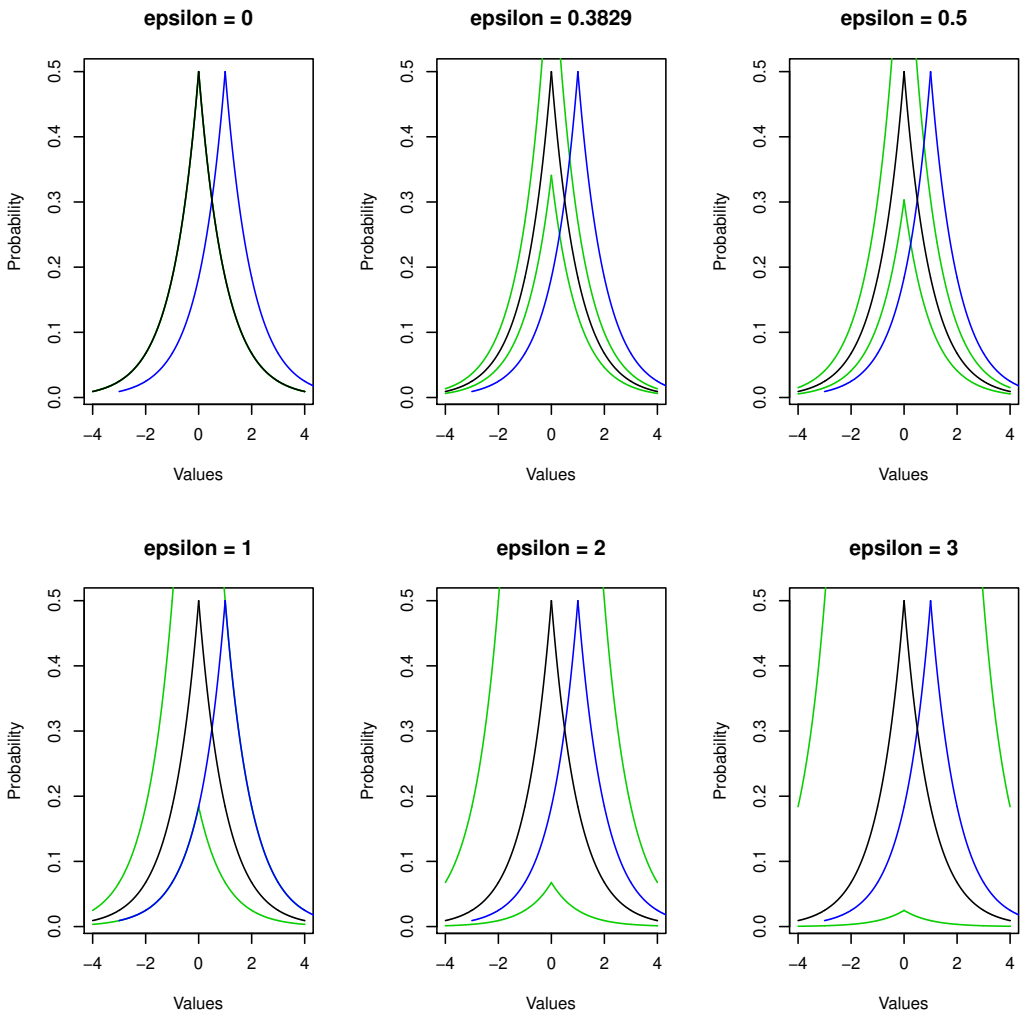
  - Let us compare different ϵ for noise following $L(0, 1)$ . . .

Differential privacy



Differential privacy

Is $0 + 1$ acceptable? I.e., are distributions L(0,1) L(1,1) similar enough?



Differential privacy

- These examples use the Laplace distribution $L(\mu, b)$.
  - I.e., probability density function:

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

    where
    - ▷ μ: location parameter
    - ▷ b: scale parameter (with $b > 0$)

- Properties
  - When $b = 1$, the function for $x > 0$ corresponds to the exponential distribution scaled by $1/2$.
  - Laplace has fatter tails than the normal distribution
  - When $\mu = 0$, for all translations $z \in \mathbb{R}$, $h(x+z)/h(x) \leq exp(|z|)$.

Differential privacy for numerical data: appropriate noise

Differential privacy

- Implementation of differential privacy for a numerical query.
  - $K_q(D)$ is a randomized version of $q(D)$:

$$K_q(D) = q(D) + \textit{and some appropriate noise}$$

  - What is $\textit{and some appropriate noise}$?

Differential privacy

- Implementation of differential privacy for a numerical query.
- Sensitivity of a query
  - Let $\mathcal{D}$ denote the space of all databases; let $q : \mathcal{D} \to \mathbb{R}^d$ be a query; then, the sensitivity of q is defined

$$\Delta_{\mathcal{D}}(q) = \max_{D,D' \in \mathcal{D}} ||q(D) - q(D')||_1.$$

where $||\cdot||_1$ is the L_1 norm, that is, $||(a_1, \ldots, a_d)||_1 = \sum_{i=1}^{d} |a_i|$.

Differential privacy

- Sensitivity of a query
  - Definition essentially meaningful when data has upper & lower bounds

An example: the case of the mean

Differential privacy

- Implementation of differential privacy: The case of the mean.
  - Sensitivity of the mean:

$$\Delta_{\mathcal{D}}(mean) = (max - min)/S$$

where $[min, max]$ is the range of the attribute, and S is the minimal cardinality of the set.

▷ If no assumption is made on the size of S: $\Delta_{\mathcal{D}}(mean) = (max - min)$

Differential privacy

- Implementation of differential privacy: The case of the mean.
  - Sensitivity of the mean:

$$\Delta_{\mathcal{D}}(mean) = (max - min)/S$$

where $[min, max]$ is the range of the attribute, and S is the minimal cardinality of the set.

▷ If no assumption is made on the size of S: $\Delta_{\mathcal{D}}(mean) = (max - min)$

  - **Proof.** Assume $S - 1$ values all in one extreme of the $[min, max]$ interval (say, max) and then we add one in the other extreme of the interval (say, min). Then, difference between the means is

$$\frac{(S-1)\cdot max}{S-1} - \frac{(S-1)\cdot max + min}{S} = max - \frac{(S-1)\cdot max}{S} - \frac{min}{S}$$

$$= \frac{S\cdot min - (S-1)\cdot max - min}{S} = \frac{max - min}{S}$$

Differential privacy

- Implementation of differential privacy for a numerical query.
  - Differential privacy via noise addition to the true response
  - Noise following a Laplace distribution $L(0, b)$ with mean equal to zero and scale parameter $b = \Delta(q)/\epsilon$.
    ($\Delta(q)$ is the sensitivity of the query)

- **Theorem.** The Laplace mechanism satisfies ϵ-differential privacy (proof Section 5.1.1, also later here)

Differential privacy

- Implementation of the Laplace mechanism

Algorithm Differential privacy for a numerical response $LM(D, q, \epsilon)$

Data: D: Database; q: query; ϵ: parameter of differential privacy

Result: Answer to the query q satisfying ϵ-differential privacy

begin

$a := q(D)$ with the original data

Compute $\Delta_{\mathcal{D}}(q)$, the sensitivity of the query for a space of databases D

Generate a random noise r from a $L(0, b)$ where $b = \Delta(q)/\epsilon$

return $a + r$

end

Differential privacy

- **Def. 3.17.** A function K_q for a query q gives (ϵ, δ)-differential privacy if for all data sets D_1 and D_2 differing in at most one element, and all $S \subseteq Range(K_q)$,

$$Pr[K_q(D_1) \in S] \leq e^{\epsilon} Pr[K_q(D_2) \in S] + \delta.$$

  - Interpretation. It relaxes ϵ-DP as events with a probability smaller than δ for D_1 are still permited even if they do not occur in D_2.

- **Prop. 5.1.** Algorithm 14 replacing the expression for b above by

$$b = \frac{\Delta(q)}{\epsilon - \log(1-\delta)}$$

satisfies (ϵ, δ)-differential privacy, for $\Delta(q)$ being the sensitivity of function q.

Differential privacy

- Example. $\Delta = 1$, $\epsilon = 0.5$
  - $\delta = 0.1$
    - $b = 1/(0.5 - log(0.9)) = 1.651908$
    - But, with $\epsilon - DP$, we only need $b = 1/0.5 = 2$

An example: the case of the mean

now with numbers

Differential privacy

- Implementation of differential privacy: The case of the mean.
  - Example[2]:
    - $D = \{1000, 2000, 3000, 2000, 1000, 6000, 2000, 10000, 2000, 4000\}$
      $\Rightarrow$ mean $= 3300$
    - Adding Ms. Rich's salary 100,000 Eur/month: mean $= 12090{,}90$!
      (a extremely high salary changes the mean significantly)
      $\Rightarrow$ We infer Ms. Rich from Town was attending the unit
  - $\Rightarrow$ Differential privacy to solve this problem

[2] Average wage in Ireland (2018): 38878 $\Rightarrow$ monthly 3239 Eur
https://www.frsrecruitment.com/blog/market-insights/average-wage-in-ireland/

Differential privacy

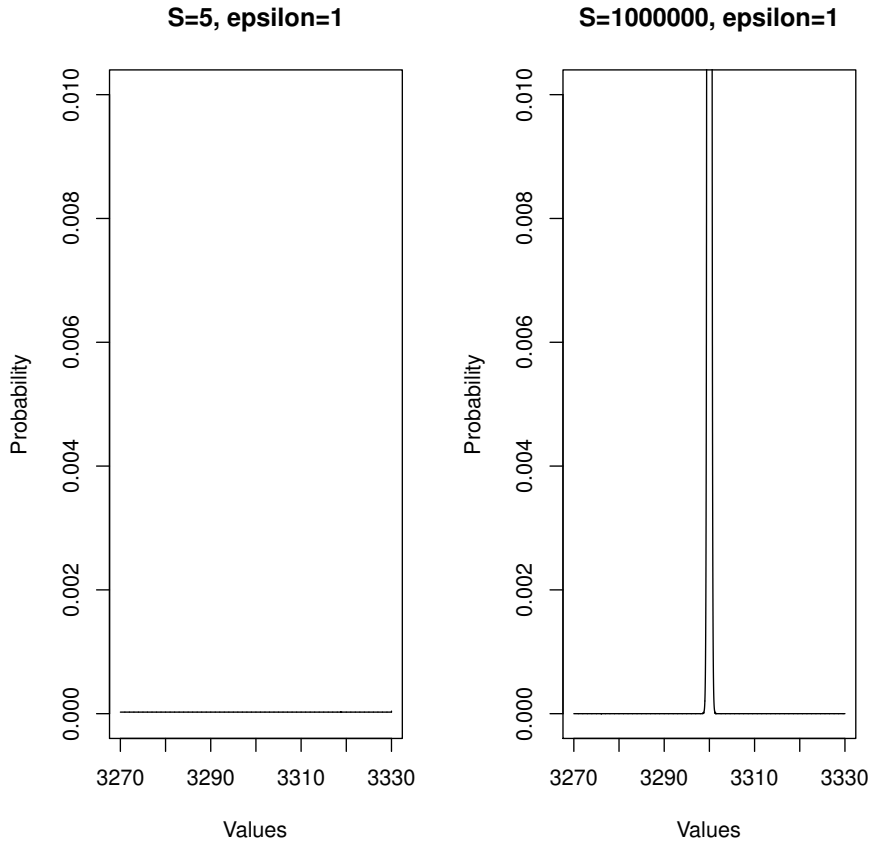
- Implementation of differential privacy: The case of the mean
  - Consider the mean salary
  - Range of salaries $[1000, 100000]$

Differential privacy

- Implementation of differential privacy: The case of the mean
  - Consider the mean salary
  - Range of salaries $[1000, 100000]$
- Compute for $\epsilon = 1$, assume that at least $S = 5$ records
  - sensitivity $\Delta_{\mathcal{D}}(q) = (max - min)/S = 19800$
  - scale parameter $b = 19800/1 = 19800$
  - For the database: (mean = 3300)
    D={1000, 2000, 3000, 2000, 1000, 6000, 2000, 10000, 2000, 4000}
  - Output: $K_{mean}(D) = 3300 + L(0, 19800)$
- Compute for $\epsilon = 1$, assume that at least $S = 10^6$ records
  - sensitivity $\Delta_{\mathcal{D}}(q) = (max - min)/S = 0.099$
  - scale parameter $b = 0.099/1 = 0.099$
  - For the database: (mean = 3300)
    D={1000, 2000, 3000, 2000, 1000, 6000, 2000, 10000, 2000, 4000}
  - Output: $K_{mean}(D) = 3300 + L(0, 0.099)$

Differential privacy

- Comparing
  - (i) $(S = 5, \epsilon = 1)$ $K_{mean}(D) = 3300 + L(0, 19800)$ and
  - (ii) $(S = 10^6, \epsilon = 1)$ $K_{mean}(D) = 3300 + L(0, 0.099)$



Differential privacy

- Laplace mechanism for differential privacy (numerical query)

$$K_q(D) = q(D) + L(0, \Delta(q)/\epsilon)$$

  - **Proposition.** For any function q, the Laplace mechanism satisfies ϵ-differential privacy.
    - **Proof.** Let $X \sim L(0, \Delta(q)/\epsilon)$, then the probability that the output is r for D is

$$\begin{aligned} Pr(K_q(D) = r) &= Pr(q(D) + X = r) = Pr(X = r - q(D)) \\ &= L(0, b)(r - q(D)) = \frac{1}{2b} exp\left(-\frac{|r - q(D)|}{b}\right) \end{aligned}$$

    - Similarly for D':

$$Pr(K_q(D') = r) = \cdots = \frac{1}{2b} exp\left(-\frac{|r - q(D')|}{b}\right)$$

- Now,

$$\frac{Pr(K_q(D) = r)}{Pr(K_q(D') = r)} = \frac{exp\left(-\frac{|r-q(D)|}{b}\right)}{exp\left(-\frac{|r-q(D')|}{b}\right)} = exp\left(\frac{|r - q(D')| - |r - q(D)|}{b}\right)$$

as $|a| - |b| \leq |a - b|$ (triangle inequality)

$$exp\left(\frac{|r - q(D')| - |r - q(D)|}{b}\right) \leq exp(\frac{|q(D) - q(D')|}{b}).$$

- As $\Delta_{\mathcal{D}}(q) = \max_{D,D' \in \mathcal{D}} ||q(D) - q(D')||_1$, then, $\Delta_{\mathcal{D}}(q) \geq ||q(D) - q(D')||_1$ for a pair of neighbouring D, D'. Therefore

$$exp\left(\frac{|q(D) - q(D')|}{b}\right) \leq exp\left(\frac{\Delta_{\mathcal{D}}(q)}{b}\right) = exp\left(\frac{\Delta_{\mathcal{D}}(q)}{\Delta_{\mathcal{D}}(q)/\epsilon}\right) = exp(\epsilon)$$

□

Means: Bounded and truncated

Differential privacy

- If the range of values is large, and sensitivity is large
  - Can we reduce the amount of noise?

Differential privacy: truncated mean

- Let us assume that the outcome is in the range $[mn, mx]$ (with $mn \neq mx$)
- Sensitivity is reduced: it is at most $[mx, mn]$
- Revisit the function mean, forcing output to be in $[mx, mn]$
  - We use

$$q'_{mn,mx}(x) = \begin{cases} mn & \text{if } x < mn \\ x & \text{if } mn \leq x \leq mx \\ mx & \text{if } mx < x \end{cases}$$

  - Then, define $q(D) = mean(D)$, and
  - $\tilde{q}(D) = q'_{mn,mx}(mean(D))$

Differential privacy

Algorithm truncated mean

Data: D: Database; S: minimum size of D; ϵ: parameter of differential privacy; mn, mx: real; max, min: real

Result: $truncated\text{-}mean$ satisfying ϵ-differential privacy and within the interval $[mn, mx]$

begin

$\Delta(mean) = \min((max - min)/S, mx - mn)$

$b = \Delta(mean)/\epsilon$

$m_0 = q'_{mn,mx}(mean(D))$ // A truncated mean

$m_1 = m_0 + L(0, b)$ // We add noise to the mean

$m_2 = q'(m_1)$ // Our output should also be in $[mn, mx]$

return (m_2)

end

Differential privacy

- Sensitivity of the truncated mean
  - If the range of the attribute is $[min, max]$,
  - S corresponds to the size of the database,
  - Sensitivity of the mean is:

$$\Delta_{\mathcal{D}}(mean) = (max - min)/S.$$

Differential privacy

- Sensitivity of the truncated mean
  - If the range of the attribute is $[min, max]$,
  - S corresponds to the size of the database,
  - Sensitivity of the mean is:

$$\Delta_{\mathcal{D}}(mean) = (max - min)/S.$$

  - But as truncated, and output in $[mn, mx]$

$$\Delta_{\mathcal{D}}(\tilde{q}) = \min((max - min)/S, (mx - mn)).$$

Differential privacy

- Sensitivity of the truncated mean. Example:
  - $[mn, mx] = [2000, 4000]$ and with $S = 5$
  - Sensitivity

$$\begin{aligned}\Delta_{\mathcal{D}}(\tilde{q}) &= \min((max - min)/5, (mx - mn)) \\ &= \min((1000000 - 1000)/5, (4000 - 2000)) = 2000.\end{aligned}$$

When we apply $\tilde{q}$ to

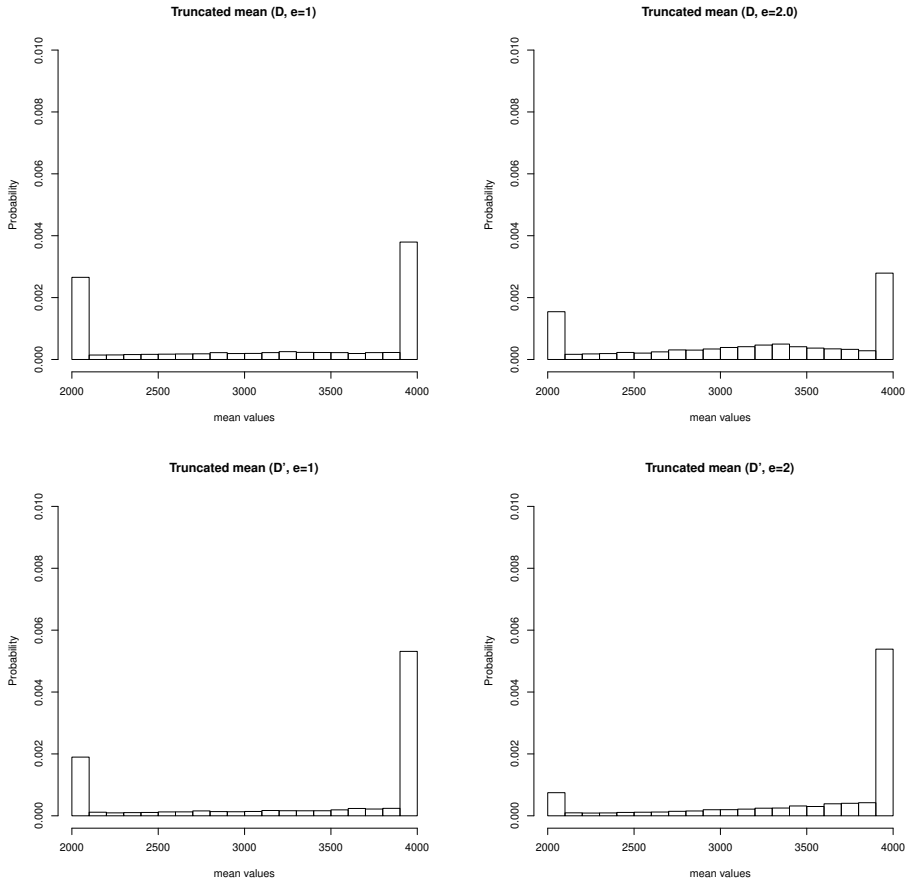
1000, 2000, 3000, 2000, 1000, 6000, 2000, 10000, 2000, 4000

we have that the real mean is $mean = 3300$.

  - So, $L(0, b)$
    - For $\epsilon = 1$, we have $b = \Delta_{\mathcal{D}}/\epsilon = 2000/1 = 2000$
    - For $\epsilon' = 0.4$, we have $b' = \Delta_{\mathcal{D}}/\epsilon' = 2000/0.4 = 5000$,
    - For $\epsilon'' = 2$, we have $b'' = \Delta_{\mathcal{D}}/\epsilon'' = 2000/2 = 1000$.

Differential privacy

- Truncated mean, $\epsilon = 1$ and $\epsilon'' = 2$
  - D and $D' = D \cup \{\text{Dona Obdúlia's}\}$ with income 1000000,
  - Figures: applying $\tilde{q}$ 10000 times.



Differential privacy

- Alternatives
  - If output is in $D = [mn, mx]$, use a bounded Laplace distribution

$$L'(x; \mu, b) = \begin{cases} 0 & \text{if } x \notin D \\ \frac{1}{C_q}\frac{1}{2b} exp\left(-\frac{|x-\mu|}{b}\right) & \text{if } x \in D \end{cases}$$

    - ▷ ϵ-DP guaranteed for sensitivity $(mx - mn)$ and $b = \Delta(q)/\epsilon$. Not otherwise.
  - Use the Laplace mechanism and re-draw the mechanism until a value in D is obtained.

Composition theorems

Differential privacy: Composition theorems

- **Sequential composition.** $q_1, \ldots, q_n$ with $\epsilon_1, \ldots, \epsilon_n$ all applied to X provide $\epsilon = \sum_{i=1}^{n} \epsilon_i$ differential privacy
  - Example #1. Apply mean and variance to X
  - Example #2. Apply mean 5 times: $\epsilon = 5 \cdot \epsilon'$
- **Parallel composition.** $q_1, \ldots, q_n$ with $\epsilon_1, \ldots, \epsilon_n$ each applied to a disjoint X_i provide $\epsilon = \max_{i=1}^{n} \epsilon_i$ differential privacy
  - Max ϵ_i means smallest protection
  - Example: mean income of different towns
- **Post-processing.** q with ϵ applied to X, and q' applied to the result of q, then $q'(q(X))$ provides ϵ differential privacy
  - Example. Compute mean, then change currency

Differential privacy

- Properties of differential privacy
  - On the ϵ:
    - Small ϵ, more privacy, more noise into the solution
    - Large ϵ, less privacy, less noise into the solution
  - On the sensitivity:
    - Small sensitivity, less noise for achieving the same privacy
    - Large sensitivity, more noise for achieving the same privacy
  - Discussion here is for a single query (with privacy budget ϵ). Multiple queries (even multiple applications of the same query) need special treatment. E.g., additional privacy budget.
  - Randomness via e.g. Laplace means that any number can be selected. Including e.g. negative ones for salaries. Special treatment may be necessary.
  - Implementations for other type of functions
    - The exponential mechanism for non-numerical queries
    - Differential privacy for machine learning and statistical models

Computing histograms

Histograms

- Histogram: frequency of a set of items for a set of buckets (or bins).
- Differential privacy: Key aspects
  - Absolutely relevant whether the set of buckets is predefined or not.
    - Buckets are defined independently of the computation of the histogram, or they are built somehow from the data.
  - $B = \{b_1, \ldots, b_b\}$ be a set of b buckets,
  - D database
  - $c_D(b_i)$ counts for each bucket $i = 1, \ldots, b$
- Given D_1 and D_2 that differ in a single record, sensitivity of the two corresponding histograms is one

Histograms

- ϵ-DP histogram
  - $c'_D(b_i) = c_D(b_i) + r_i$ with r_i following $L(0, b)$
  - $b = \Delta(q)/\epsilon = 1/\epsilon$

Histograms

- Example: ϵ-DP histogram
  - Buckets: $b_1 = [1000, 1999]$, $b_2 = [2000, 2999]$, $b_3 = [3000, 3999]$
  - Data:
    $D = \{1234, 1300, 1233, 1250, 1284, 2000, 2300, 2044, 2573,$
    $2745, 2853, 2483, 3633, 3182, 3274, 3935\}$
  - Histogram$(D) = (5, 7, 4)$
  - Draw r_1, r_2, r_3 (independently) from $L(0, b)$ with $\epsilon = 1$, so, $b = 1$
  - Say, $r = (0.7534844, -0.6143575, -1.5725160)$
  - ϵ-DP histogram:
    $histogram(B, D) = (c'_D(b_1), c'_D(b_2), c'_D(b_3)) = (5, 7, 4) + r$
    $(5.753484, 6.385643, 2.427484)$

Histograms and composition

- Use composition to compute the mean from an histogram
  - D a database,
  - $B = \{b_1, \ldots, b_b\}$ buckets with range $b_i = [b_{in}, b_{ix})$
  - $c(b_i)$ counts
  - $m(b_i)$ the mean value of the interval.

- Approximate average as follows:
  - $mean(B, histogram(B, D)) = \frac{\sum_{i=1}^{b} c(b_i)m(b_i)}{\sum_{i=1}^{b} c(b_i)}$.

- Approximate average:
  the larger the buckets, the less acurate the mean

Histograms and composition

- Use composition to compute the mean from an histogram
  - D, $b_1 = [1000, 2000)$, $b_2 = [2000, 3000)$, $b_3 = [3000, 4000)$
  - histogram $(5, 7, 4)$
  - mean:
    $$mean(B, histogram(B, D)) = \frac{\sum_{i=1}^{b} c(b_i)m(b_i)}{\sum_{i=1}^{b} c(b_i)} = \frac{5 \cdot 1500 + 7 \cdot 2500 + 4 \cdot 3500}{5+7+4}$$
  - output: 2437.5

- Compare this result with the mean of D which is 2332.688 (i.e., non-DP)

- Other discretizations, another result!

- NOTE: We are here working with the histogram, but all computations are with non-private histograms this is not ϵ-differentially private

Histograms and composition

- Now, differentially private, using composition
  - **Step 1.** Compute $c = histogram(B, D)$
  - **Step 2.** Produce a differentially private histogram c'
  - **Step 3.** $mean(B, c') = \frac{\sum_{i=1}^{b} c'(b_i)m(b_i)}{\sum_{i=1}^{b} c'(b_i)}$
- If c' is ϵ-DP, mean(c') is also ϵ-DP (composition theorems)
- Example.
  - Using DP-histogram $(5.753484, 6.385643, 2.427484)$

$$\begin{aligned} mean(B, c') &= \frac{\sum_{i=1}^{b} c'(b_i)m(b_i)}{\sum_{i=1}^{b} c'(b_i)} \\ &= \frac{5.753484 \cdot 1500 + 6.385643 \cdot 2500 + 2.427484 \cdot 3500}{5.753484 + 6.385643 + 2.427484} \\ &= 2271.67 \end{aligned}$$

Histograms and composition

- Histograms and domain
  - We can apply this approach to compute the mean for any database.
  - We need buckets to span over the whole range of incomes.
    - So, if we consider the incomes in the range [1000, 100000] as when Dona Obdúlia was in the database, we need either a large bucket (with e.g., all incomes larger than 10000) or a large number of buckets.
  - This will have effects on the output.

Differential privacy: Categorical data

Categorical data

- Categorical output: $C = \{c_1, \ldots, c_c\}$
- Differential privacy, same definition applies
  - A function K_q for a query q gives ϵ-differential privacy if for all data sets D_1 and D_2 differing in at most one element, and all $S \subseteq Range(K_q)$,

$$\frac{Pr[K_q(D_1) \in S]}{Pr[K_q(D_2) \in S]} \leq e^{\epsilon}.$$

(with 0/0=1) or, equivalently,

$$Pr[K_q(D_1) \in S] \leq e^{\epsilon} Pr[K_q(D_2) \in S].$$

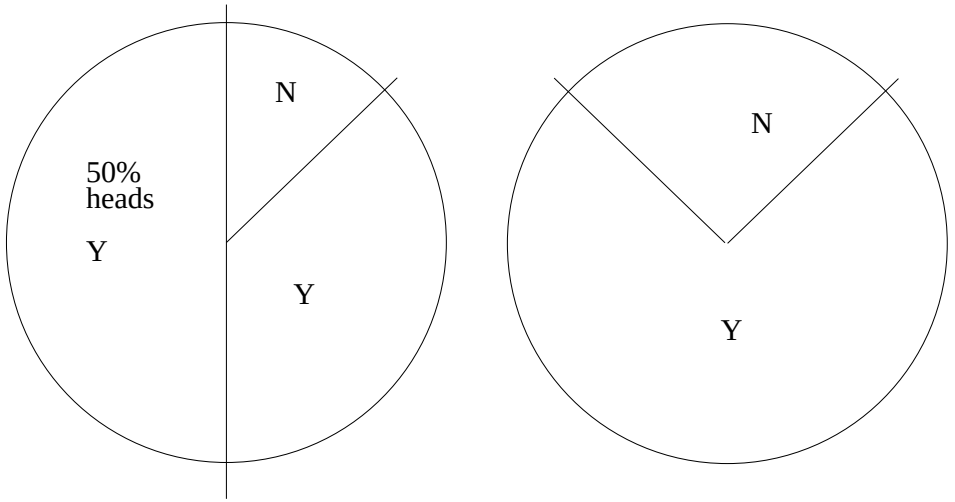
- Differential privacy using randomized response

Categorical data

- Randomized response:
  introduced for sensitive questions (Warner, 1965)
  - Categorical output, 2 outcomes: $C = \{Yes, No\}$
  - Example:
    - Have you consumed drugs this week? / Is your car now exceeding the speed limit?
  - Implementation:
    - toss a coin
    - if heads, return Yes
    - if tails, return the true answer

Categorical data

- Given all answers, we can estimate the true proportion
  - True proportion of Yes: p_N, True proportion of No: p_N.
    - Naturally, $p_Y = 1 - p_N$
    - r: proportion of answered No
    - Then, $p_N = 2 * r$, so, $p_Y = 1 - 2 * r$
  - Graphically



Categorical data: Example

- Given all answers, we can estimate the true proportion
- Example
  - We ask 100 people about their drug consumption
  - We get 45 Nos, and 55 Yes
- Answer

Categorical data: Example

- Given all answers, we can estimate the true proportion
- Example
  - We ask 100 people about their drug consumption
  - We get 45 Nos, and 55 Yes
- Answer
  - 50 answered Yes by default
  - so, $55 - 50 = 5$ answered a true yes
  - So, real yes was $2 * 5 = 10$
  - And true No was $r = 45$, So, total No is $r = 2 * 45 = 90$.

Categorical data

- In general,
  - probability p of returning right answer
  - probability p' of returning Y when false answer, $1 - p'$ of N

Categorical data

- **Algorithm** randomized response: $rr(f(X), p, p')$
  Data: $f(X)$: the true outcome of the query; p, p': probability in [0,1]

  Result: Randomized response for $f(X)$ with probabilities p, p'

  begin
    $r :=$ random number in [0,1] according to a uniform distribution
    if $r < p$ **then**
      return $f(X)$
    else
      $r' :=$ random number in [0,1] according to a uniform distribution
      if $r' < p'$ **then**
        return Y
      else
        return N
      end
    end
  end

Categorical data

- In general,
  - probability p of returning right answer
  - probability p' of returning Y when false answer, $1-p'$ of N
- π true proportion of Yes, o observed proportion of Yes

$$o = p * \pi + (1-p) * p'.$$

- So, given observed proportion o of Yes, we estimate π:

$$\hat{\pi} = (o - (1-p) * p')/p.$$

Categorical data

- In general, but with an example
  - probability $p = 0.5$ of returning right answer
  - probability $p' = 0.75$ of returning Y when false answer, $1 - p'$ of N
- We compute (assuming $\pi = 0.1$ Yes, as in the previous example)
  - π true proportion of Yes, o observed proportion of Yes

$$o = p * \pi + (1 - p) * p' \ = 1/2 * 0.1 + 1/2 * 3/4 = 0.425$$

  - So, given observed proportion o of Yes, we estimate π:

$$\hat{\pi} = (o - (1 - p) * p')/p \ = (0.425 - (1 - 0.5) * 3/4)/0.5 = 0.1$$

Categorical data

- In general, but with an example
  - probability $p = 0.5$ of returning right answer
  - probability $p' = 0.75$ of returning Y when false answer, $1 - p'$ of N
- We compute (assuming $\pi = 0.1$ Yes, as in the previous example)
  - π true proportion of Yes, o observed proportion of Yes

$$o = p * \pi + (1 - p) * p' \ = 1/2 * 0.1 + 1/2 * 3/4 = 0.425$$

  - So, given observed proportion o of Yes, we estimate π:

$$\hat{\pi} = (o - (1 - p) * p')/p \ = (0.425 - (1 - 0.5) * 3/4)/0.5 = 0.1$$

- However, in general, the larger the noise (i.e., p is very small), the more difficult to recover π: observe, $p = 0$. (Warner, 1965)

Differential privacy: general case with multiple categories

Categorical data

- General case: $C = \{c_1, \ldots, c_c\}$.
  - Randomized response, with a probability distribution for each c_i
  - from c_i to c_j

$$P(c_i, c_j) = P(X' = c_j | X = c_i).$$

Categorical data

- General case: $C = \{c_1, \ldots, c_c\}$.
  - Randomized response, with a probability distribution for each c_i
  - from c_i to c_j

$$P(c_i, c_j) = P(X' = c_j | X = c_i).$$

  - Naturally, for each c_i we have

$$P(X' = c_1 | X = c_i), \ldots, P(X' = c_c | X = c_i)$$

for all c_i it holds $\sum_j P(c_i, c_j) = \sum_j P(X' = c_j | X = c_i) = 1$.

Categorical data

- General case: $C = \{c_1, \ldots, c_c\}$.
  - Randomized response, with a probability distribution for each c_i
  - from c_i to c_j

$$P(c_i, c_j) = P(X' = c_j | X = c_i).$$

  - Naturally, for each c_i we have

$$P(X' = c_1 | X = c_i), \ldots, P(X' = c_c | X = c_i)$$

    for all c_i it holds $\sum_j P(c_i, c_j) = \sum_j P(X' = c_j | X = c_i) = 1$.
  - $P(c_i, c_j)$ a transition matrix P where the rows add to one
  - This is (like) PRAM

Categorical data

- General case: $C = \{c_1, \ldots, c_c\}$.

- **Algorithm** randomized response via PRAM: $rrPRAM(c, P)$
  Data: c: the true outcome of the query; P: transition matrix

  Result: Randomized response for c according to transition matrix P

  begin

  $r :=$ random number in [0,1] according to a uniform distribution
  Select k_0 in $\{1, \ldots, c\}$ such that
  $$\sum_{k=1}^{k_0-1} P(c' = c_i, |C = c) < r \leq \sum_{k=1}^{k_0} P(c' = c_i, |C = c)$$
  return c_{k_0}

  end

Categorical data

- General case: $C = \{c_1, \ldots, c_c\}$, true proportions?
  - After protection we observe: $o = (o_1, \ldots, o_c)$
  - but, the true response was $\pi = (\pi_1, \ldots, \pi_c)$
    here π_k is the proportion of respondents of class c_k
  - How to compute π from o?

Categorical data

- General case: $C = \{c_1, \ldots, c_c\}$, true proportions?
  - After protection we observe: $o = (o_1, \ldots, o_c)$
  - but, the true response was $\pi = (\pi_1, \ldots, \pi_c)$
    here π_k is the proportion of respondents of class c_k
  - How to compute π from o?
  - We know o from π:

$$o_j = \sum_{i=1}^{c} \pi_i P(X' = c_j | X = c_i)$$

    in matrix form:

$$o = P\pi$$

  - So, we can estimate

$$\hat{\pi} = P^{-1} o$$

Categorical data

- Randomized response = PRAM
  - The approach discussed here corresponds to PRAM
  - While PRAM assumes that we have the database available, Randomized response often considers local data being transmitted

Categorical data

- Randomized response = PRAM
  - The approach discussed here corresponds to PRAM
  - While PRAM assumes that we have the database available, Randomized response often considers local data being transmitted
  - i.e., local differential privacy

Appropriate noise: categorical data

Categorical data

- Local differential privacy, reminder, and rewriting
  - D_1 and D_2 are single records or categories

$$\frac{Pr[K_q(D_1) \in S]}{Pr[K_q(D_2) \in S]} \leq e^{\epsilon}.$$

Categorical data

- Local differential privacy, reminder, and rewriting
  - D_1 and D_2 are single records or categories

$$\frac{Pr[K_q(D_1) \in S]}{Pr[K_q(D_2) \in S]} \leq e^{\epsilon}.$$

  - with categories

$$\frac{Pr[K_q(c_i) = c_c]}{Pr[K_q(c_j) = c_c]} \leq e^{\epsilon}.$$

Categorical data

- Local differential privacy, reminder, and rewriting
  - D_1 and D_2 are single records or categories

$$\frac{Pr[K_q(D_1) \in S]}{Pr[K_q(D_2) \in S]} \leq e^{\epsilon}.$$

  - with categories

$$\frac{Pr[K_q(c_i) = c_c]}{Pr[K_q(c_j) = c_c]} \leq e^{\epsilon}.$$

  - and, in PRAM-like / randomized-response like

$$\frac{P(X' = c_c|c_i)}{P(X' = c_c|c_j)} \leq e^{\epsilon}.$$

Categorical data

- What is the appropriate noise ? (given ϵ)
- Assumptions on the matrix:
  - All categories same probability of being modified
    for all c_i, c_j we have $P(X' = c_i|c_i) = P(X' = c_j|c_j)$.
  - Non-diagonal values are all equal
    $P(X' = c_i|c_j) = P(X' = c_k|c_l)$ for all $i \neq j$, $k \neq l$
  - We assume $P(X' = c_i|c_i) > P(X' = c_j|c_i)$ for all $i \neq j$
- Summary, matrix of this form

$$\begin{pmatrix} q_d & q & \dots & q \\ q & q_d & \dots & q \\ \dots & & & \dots \\ q & q & \dots & q_d \end{pmatrix} \quad (1)$$

with $q_d = P(X' = c_i|c_i)$ for all i, and $q = P(X' = c_j|c_i)$ for $j \neq i$.

Categorical data

- What is the appropriate noise ? (given ϵ)
  - Probabilities after masking, we had c_i ?
    $$(P(X' = c_1|c_i), ..., P(X' = c_c|c_i)).$$
  - Probabilities after masking, we had c_j ?
    $$(P(X' = c_1|c_j), ..., P(X' = c_c|c_j)).$$

Categorical data

- What is the appropriate noise ? (given ϵ)
  - Probabilities after masking, we had c_i ?
    $$(P(X' = c_1|c_i), ..., P(X' = c_c|c_i)).$$
  - Probabilities after masking, we had c_j ?
    $$(P(X' = c_1|c_j), ..., P(X' = c_c|c_j)).$$
  - We require local ϵ-differential privacy, so we need
    $$P(X' = c_1|c_i)/P(X' = c_1|c_j) \leq e^\epsilon, \ldots, P(X' = c_c|c_i)/P(X' = c_c|c_j)) \leq e^\epsilon,$$
    This means
    $$\max_{k=1}^{c} P(X' = c_k|c_i)/P(X' = c_k|c_j) \leq e^\epsilon$$

Categorical data

- What is the appropriate noise ? (given ϵ)
  - Probabilities after masking, we had c_i ?
    $$(P(X' = c_1|c_i), ..., P(X' = c_c|c_i)).$$
  - Probabilities after masking, we had c_j ?
    $$(P(X' = c_1|c_j), ..., P(X' = c_c|c_j)).$$
  - We require local ϵ-differential privacy, so we need
    $$P(X' = c_1|c_i)/P(X' = c_1|c_j) \leq e^\epsilon, \ldots, P(X' = c_c|c_i)/P(X' = c_c|c_j)) \leq e^\epsilon,$$
    This means
    $$\max_{k=1}^{c} P(X' = c_k|c_i)/P(X' = c_k|c_j) \leq e^\epsilon$$
  - We assumed $P(X' = c_i|c_i)$ the largest value in a row, and all non-diagonal values are the same, so, maximum is obtained for $k = i$.

Categorical data

- What is the appropriate noise ? (given ϵ)
  - Probabilities after masking, we had c_i ?
    $$(P(X' = c_1|c_i), ..., P(X' = c_c|c_i)).$$
  - Probabilities after masking, we had c_j ?
    $$(P(X' = c_1|c_j), ..., P(X' = c_c|c_j)).$$
  - We require local ϵ-differential privacy, so we need
    $$P(X' = c_1|c_i)/P(X' = c_1|c_j) \leq e^\epsilon, \ldots, P(X' = c_c|c_i)/P(X' = c_c|c_j)) \leq e^\epsilon,$$
    This means
    $$\max_{k=1}^{c} P(X' = c_k|c_i)/P(X' = c_k|c_j) \leq e^\epsilon$$
  - We assumed $P(X' = c_i|c_i)$ the largest value in a row,
    and all non-diagonal values are the same,
    so, maximum is obtained for $k = i$.
    In order to get precisely ϵ privacy (and not $\epsilon_0 < \epsilon$ privacy) we require the equality to hold.
    $$P(X' = c_i|c_i)/P(X' = c_i|c_j) = e^\epsilon. \quad (2)$$

Categorical data

- What is the appropriate noise ? (given ϵ)
  - From these quotients, we compute the values, how?
    $$P(X' = c_i|c_i)/P(X' = c_i|c_j) = e^{\epsilon}.$$
  - each row needs to add to one, so
    $$P(X' = c_i|c_i) + (c-1)P(X' = c_i|c_j) = 1,$$
    or, equivalently, $P(X' = c_i|c_j) = (1 - P(X' = c_i|c_i))/(c-1)$.
  - Using this expression, we have that Equation 2 becomes
    $$P(X' = c_i|c_i)/((1 - P(X' = c_i|c_i))/(c-1)) = e^{\epsilon}.$$
  - This equality implies that $P(X' = c_i|c_i)$ is of the following form:
    $$P(X' = c_i|c_i) = e^{\epsilon}/(c - 1 + e^{\epsilon}),$$
  - and, therefore, $P(X' = c_i|c_j)$ for $i \neq j$ is
    $$P(X' = c_i|c_j) = 1/(c - 1 + e^{\epsilon}).$$

Categorical data

- Example: What is the appropriate noise? (given ϵ, and $c = 2$)
  - Our matrix will have this form

$$\begin{pmatrix} \frac{e^\epsilon}{1+e^\epsilon} & \frac{1}{1+e^\epsilon} \\ \frac{1}{1+e^\epsilon} & \frac{e^\epsilon}{1+e^\epsilon} \end{pmatrix} \tag{3}$$

Categorical data

- What is the appropriate noise? (given ϵ)
- Example. Maximum privacy $c = 2$ and $\epsilon = 0$,
  - the transition matrix contains only $1/2$.

$$\begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}$$

Categorical data

- What is the appropriate noise? (given ϵ, and $c = 2$)
  - Example. $c = 2$ and $\epsilon = 1$
    Answers: {I like this app, I do not like this app}

$$\begin{pmatrix} \frac{e^\epsilon}{1+e^\epsilon} = 0.73 & \frac{1}{1+e^\epsilon} = 0.27 \\ \frac{1}{1+e^\epsilon} = 0.27 & \frac{e^\epsilon}{1+e^\epsilon} = 0.73 \end{pmatrix}$$

Categorical data

- What is the appropriate noise? (given ϵ, and $c = 2$)
  - Example. $c = 2$ and $\epsilon = 10$
    Answers: {I like this app, I do not like this app}

$$\begin{pmatrix} \frac{e^\epsilon}{1+e^\epsilon} = 0.9999546 & \frac{1}{1+e^\epsilon} = 0.000123 \\ \frac{1}{1+e^\epsilon} = 0.000123 & \frac{e^\epsilon}{1+e^\epsilon} = 0.9999546 \end{pmatrix}$$

Categorical data

- What is the appropriate noise? (given ϵ, and $c = 2$)
  - Example. $c = 7$ and $\epsilon = 10$
  - Answers (Do you like this app): {Not-at-all, don't, ..., fantastic}

$$\begin{pmatrix} \frac{e^\epsilon}{c-1+e^\epsilon} = 0.9997277 & \dots & \frac{1}{c-1+e^\epsilon} = 0.0001233 & \frac{1}{c-1+e^\epsilon} = 0.000123 \\ \vdots & & \vdots & \vdots \\ \vdots & & \vdots & \vdots \\ \vdots & & \vdots & \vdots \\ \frac{1}{c-1+e^\epsilon} = 0.0001233 & \dots & \frac{1}{c-1+e^\epsilon} = 0.0001233 & \frac{e^\epsilon}{c-1+e^\epsilon} = 0.9997277 \end{pmatrix}$$

Categorical data

- Discussion based on Assumptions on the matrix (i, j, k as above)
  - $q_d = P(X' = c_i|c_i) = P(X' = c_j|c_j)$ (diagonal)
  - $q = P(X' = c_i|c_j) = P(X' = c_k|c_l)$
  - and $q_d = P(X' = c_i|c_i) > P(X' = c_j|c_i) = q$
- Nevertheless, we may have other assumptions

Categorical data

- Discussion based on Assumptions on the matrix (i, j, k as above)
  - $q_d = P(X' = c_i|c_i) = P(X' = c_j|c_j)$ (diagonal)
  - $q = P(X' = c_i|c_j) = P(X' = c_k|c_l)$
  - and $q_d = P(X' = c_i|c_i) > P(X' = c_j|c_i) = q$

- Nevertheless, we may have other assumptions
  - When $c = 2$, most general case (assume outputs 0 and 1)

$$\begin{pmatrix} p_{00} & p_{01} = 1 - p_{00} \\ p_{10} = 1 - p_{11} & p_{11} \end{pmatrix} \tag{4}$$

Categorical data

- Most general case $c = 2$
  - Matrix

$$\begin{pmatrix} p_{00} & p_{01} = 1 - p_{00} \\ p_{10} = 1 - p_{11} & p_{11} \end{pmatrix} \tag{5}$$

  - Region of feasibility (i.e., possible p_{00} and p_{11})

Categorical data

- Most general case $c = 2$
  - Matrix

$$\begin{pmatrix} p_{00} & p_{01} = 1 - p_{00} \\ p_{10} = 1 - p_{11} & p_{11} \end{pmatrix} \tag{5}$$

  - Region of feasibility (i.e., possible p_{00} and p_{11})
    - $p_{00} \leq (1 - p_{11})e^{\epsilon}$
    - $p_{11} \leq (1 - p_{00})e^{\epsilon}$
    - $(1 - p_{00}) \leq p_{11}e^{\epsilon}$
    - $(1 - p_{11}) \leq p_{00}e^{\epsilon}$

Categorical data

- Local differential privacy

Categorical data

- Local differential privacy
  - Multiple release, non-independent, series of ith sensor at time t

$$x_i^1, \ldots, x_i^T$$

  - protected using ρ (e.g., ϵ-DP)

$$\rho(x_i^1), \ldots, \rho(x_i^T)$$

Categorical data

- Local differential privacy
  - Multiple release, non-independent, series of ith sensor at time t

$$x_i^1, \ldots, x_i^T$$

  - protected using ρ (e.g., ϵ-DP)

$$\rho(x_i^1), \ldots, \rho(x_i^T)$$

  - if ϵ the whole budget, we need ϵ/T-DP for ρ

In the other way round: ϵ-DP for PRAM

Privacy guarantees

- Given a matrix P, what ϵ_0-DP are we providing ?
  - We already have seen that for all i and j,

$$\max_{k=1}^{c} P(X' = c_k|c_i)/P(X' = c_k|c_j) \leq e^{\epsilon}.$$

  - So, equivalently

$$\max_{i=1}^{c} \max_{j=1}^{c} \max_{k=1}^{c} P(X' = c_k|c_i)/P(X' = c_k|c_j) \leq e^{\epsilon}.$$

  - Most fitted ϵ is with the equality, so, then

$$\epsilon_0 = log\left(\max_{i=1}^{c} \max_{j=1}^{c} \max_{k=1}^{c} P(X' = c_k|c_i)/P(X' = c_k|c_j)\right).$$

- NOTE: No-zero probabilities in the matrix, otherwise ϵ_0 is infinite, no privacy guarantee from a DP point of view.

- Connection with intersection attack! (also discussed)

Neither categorical nor numerical:

Deep learning

- Deep learning is usually implemented with Stochastic Gradient Descent
  - Iterative process with (i is a sample)

$$w_{t+1} = w_t - \alpha_t \nabla g_i(w_t, x_i)$$

  - $\nabla g_i(w_t, x_i)$ is a vector of numbers, so we can just add noise

$$\nabla' g_i(w_t, x_i) = \nabla g_i(w_t, x_i) + Lap(\Delta(\nabla g)/\epsilon)$$

- Problems
  - The amount of noise is too high, and
  - We need to do multiple iterations with lots of samples x_i

- So, we need some variations
  - norm clipping: if the norm of the vector is too large, clip it

$$||g(x)||_2 = \begin{cases} ||g(x)||_2 & ||g(x)||_2 \leq C \\ C & ||g(x)||_2 > C \end{cases}$$

  - Grouping batches: compute average gradients of a batch $x_i \in I$

Some examples with more complex data

>

- Privacy for complex data[3]
  - Privacy for graphs
  - Smart grid data
  - Streaming data, multiple releases, etc. (temporal component)
- Privacy-preserving solutions in different environments
  - Federated learning
  - Privacy models for voting and decision making

[3] This relates to our own research:
https://www.umu.se/forskning/grupper/nausica-privacy-aware-transparent-decisions-group

Privacy for graphs

Problem

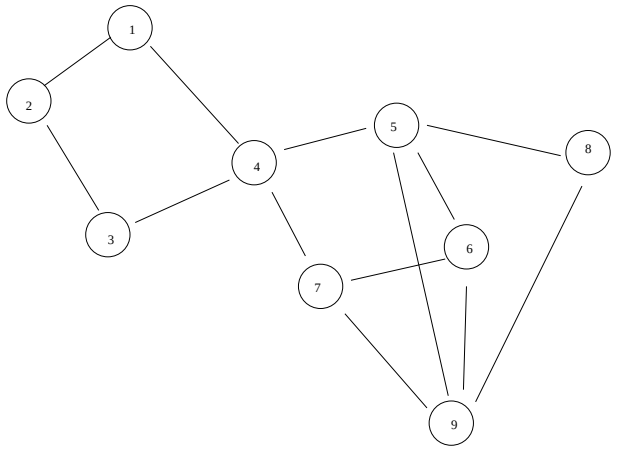
Graphs

Graph: Representation of a large number of problems

Representation:

- $G(V, E)$
  with V vertices / nodes
  with E edges $E \subseteq V \times V$

E represented by the adjacency matrix



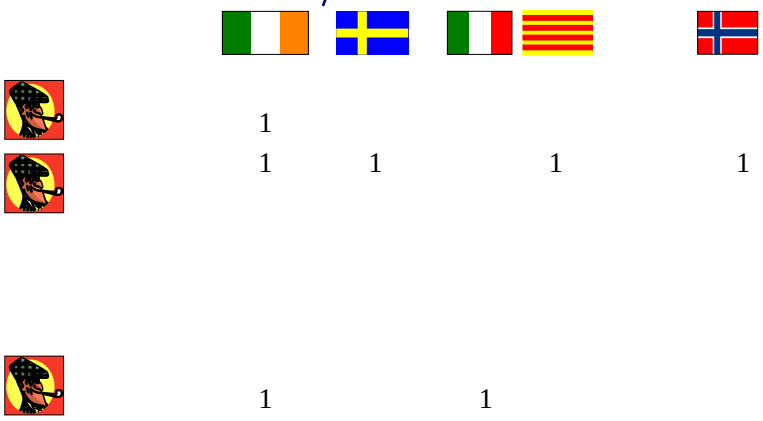
$$\begin{pmatrix} 010100000 \\ 101000000 \\ 010100000 \\ 101010100 \\ 000101011 \\ 000010101 \\ 000101001 \\ 000010001 \\ 000011110 \end{pmatrix}$$

Graphs

Graph: Representation of a large number of problems

Examples:

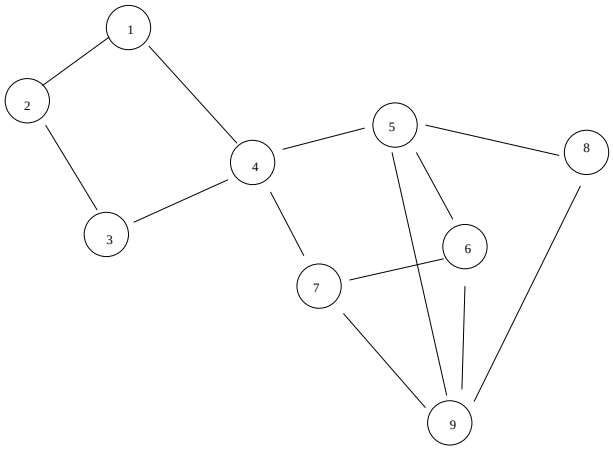
- Social networks: nodes people, edges friendships
- Real networks: topology of a communication network
- But also,
  Preferences/likes: travellers vs. countries; customers vs. products

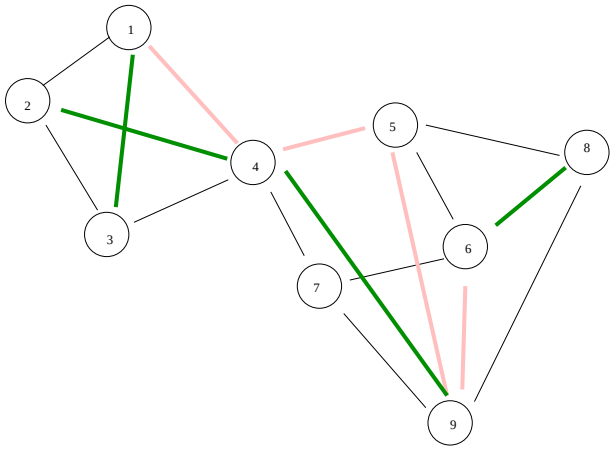


Problem

Data protection for graphs:

- Given a graph G, produce a protected graph G'
- G' ressembles G
- and avoids disclosure (e.g., do not find you)





Privacy-preserving graphs

Graphs

- Same questions as usual
  - Which is the appropriate privacy model (masking)
  - Is the data = graph useful after protection (IL)
  - Is the data = graph safe after protection (DR)

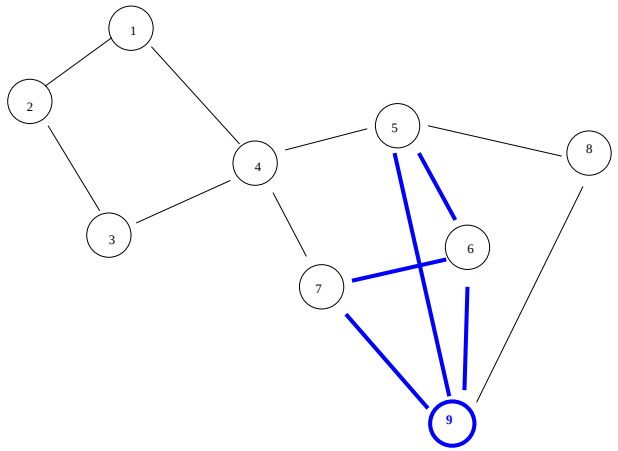
Disclosure risk

Data protection for graphs: Avoids disclosure (definition)

- An intruder with some information I on node v of the graph
- is not able to identify the node.

Example of information I

- The degree of a node (i.e., $|N(v)|$)
- The subgraph of neighbours (i.e., $\tilde{G}$ from v and $N(v)$) (subgraph isomorphism problem // subgraph matching)



Disclosure risk and privacy models

Data protection for graphs: Avoids disclosure (definition)

- An intruder with some information I on node v of the graph
- is not able to identify the node.

Privacy models for graphs

- k-anonymity for graphs

Disclosure risk and privacy models

Data protection for graphs: Avoids disclosure (definition)

- An intruder with some information I on node v of the graph
- is not able to identify the node.

Privacy models for graphs

- k-anonymity for graphs
  - k-degree anonymity

Disclosure risk and privacy models

Data protection for graphs: Avoids disclosure (definition)

- An intruder with some information I on node v of the graph
- is not able to identify the node.

Privacy models for graphs

- k-anonymity for graphs
  - k-degree anonymity
  - 1-neihborhood anonymity, for each node n there are $k-1$ other nodes $n_1, \ldots, n_{k-1}$ such that the graphs $\mathcal{N}(n_i)$ are isomorphic

Disclosure risk and privacy models

Data protection for graphs: Avoids disclosure (definition)

- An intruder with some information I on node v of the graph
- is not able to identify the node.

Privacy models for graphs

- k-anonymity for graphs
  - k-degree anonymity
  - 1-neihborhood anonymity, for each node n there are $k-1$ other nodes $n_1, \ldots, n_{k-1}$ such that the graphs $\mathcal{N}(n_i)$ are isomorphic
- Differential privacy
  - Node and degree-differential privacy

k-anonymity

Data protection: k-degree anonymity

- Given G produce G' (k-degree anonymity)
  - Find degree sequence of G
  - Microaggregate degree sequence (integer! degree sequence)
  - Degree sequence needs to be graphical!!
  - Swap edges in graph G to achieve G' with appropriate degree

Data protection: k-anonymity

- Given $G = (V, E)$ produce G' (k-degree anonymity)
  - Define $G' = (V', E')$ with the nodes $n \in V$ in G
    $V' = V$

Data protection: k-anonymity

- Given $G = (V, E)$ produce G' (k-degree anonymity)
  - Define $G' = (V', E')$ with the nodes $n \in V$ in G
    $V' = V$
  - Find clusters of nodes of size at least k
    $c(n)$ is the cluster associated to n

Data protection: k-anonymity

- Given $G = (V, E)$ produce G' (k-degree anonymity)
  - Define $G' = (V', E')$ with the nodes $n \in V$ in G
    $V' = V$
  - Find clusters of nodes of size at least k
    $c(n)$ is the cluster associated to n
  - Decide whether cluster i and cluster j are $connected$ or not
    (for all i, j)

Data protection: k-anonymity

- Given $G = (V, E)$ produce G' (k-degree anonymity)
  - Define $G' = (V', E')$ with the nodes $n \in V$ in G
    $V' = V$
  - Find clusters of nodes of size at least k
    $c(n)$ is the cluster associated to n
  - Decide whether cluster i and cluster j are $connected$ or not (for all i, j)
  - For each node n_1 and n_2, define E' as follows
    $edge(n_1, n_2) = 1$ if and only if $c(n_1)$ is connected to $c(n_2)$

Differential privacy

Data protection: edge-differential privacy

- Given G produce G' (ϵ-edge differential privacy)
  - An edge randomization algorithm $\mathcal{A} : \mathcal{G} \to \mathcal{G}$, satisfies ϵ-edge local differential privacy if for every pair of nodes $u, v \in V$ and $x, x', y \in \{0, 1\}$:

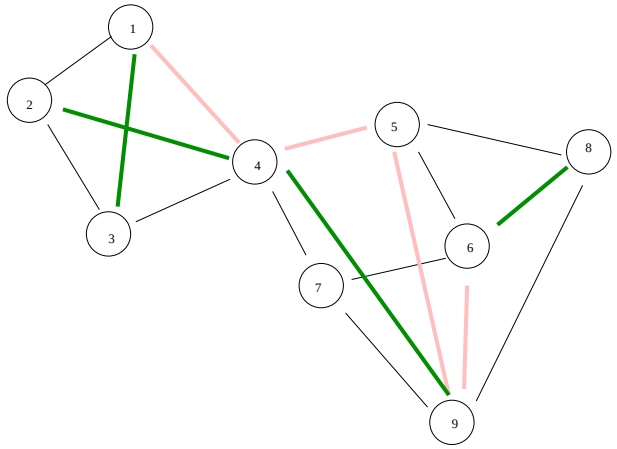
$$P(\mathbb{1}_{\mathcal{A}(uv)} = y | \mathbb{1}_{uv} = x) \leq e^{\epsilon} P(\mathbb{1}_{\mathcal{A}(uv)}) = y | \mathbb{1}_{uv} = x'),$$

  we say that $\mathcal{A}$ is ϵ-edge locally differentially private

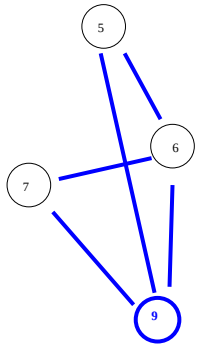
Data protection

Data protection for graphs: How to ?

- Adhoc protection: change structure
  - Random addition and deletion of nodes
  - Random addition and deletion of edges
  - Check how much addition / deletion is needed with some attacks



Find this now?



Graph addition

Noise addition (for numerical data)

Our proposal:

- Inspired in noise addition for numerical data
- Add noise to hide e.g. age and salary

Noise addition: Data protection via noise addition

$$X' = X + \epsilon$$

with $\epsilon \sim N(0, kVar)$

- This definition permits to deduce properties for X'
  (e.g., mean of X' = mean of X, variance of X', etc.)
  Related definitions with correlated noise in multivariate X

Noise addition for graphs

Noise addition for graphs: Similar idea but with graphs

$$G' = G \oplus g$$

- $G \oplus g$ for $G = (V, E)$ and $g = (V_g, E_g)$ as follows
  - align nodes of both graphs
  - edges in terms of exclusive-or of edges, or symmetric difference.

$$\begin{aligned} E_1 \Delta E_2 &:= (E_1 \setminus E_2) \cup (E_2 \setminus E_1) \\ &\{e | e \in E_1 \wedge e \notin E_2\} \cup \{e | e \notin E_1 \wedge e \in E_2\} \end{aligned}$$

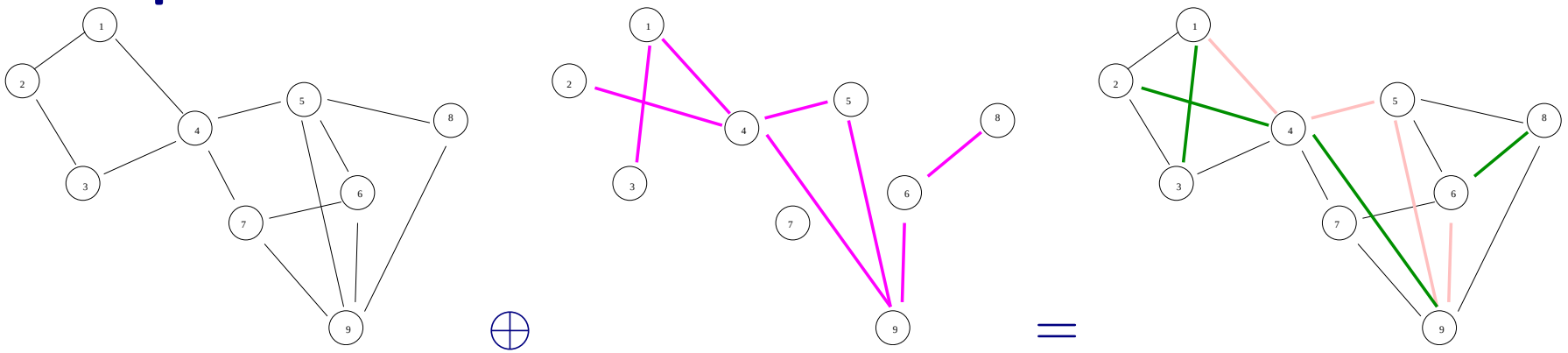
$\rightarrow G' = (V', E')$ with $E' = E \Delta E_g$

Noise addition for graphs: Example

Noise addition for graphs: Similar idea but with graphs

$$G' = G \oplus g$$

- **Example:**



Noise addition: random graphs

Noise addition for graphs: Similar idea but with graphs

$$G' = G \oplus g$$

- g is a random graph[4]

[4] VT, JS, Graph Perturbation as Noise Graph Addition: A New Perspective for Graph Anonymization. Proc. DPM 2019; JS, VT, Differentially Private Graph Publishing and Randomized Response for Collaborative Filtering. Proc. SECRYPT 2020

Noise addition: Graphs to add

Graphs. Examples of random graphs

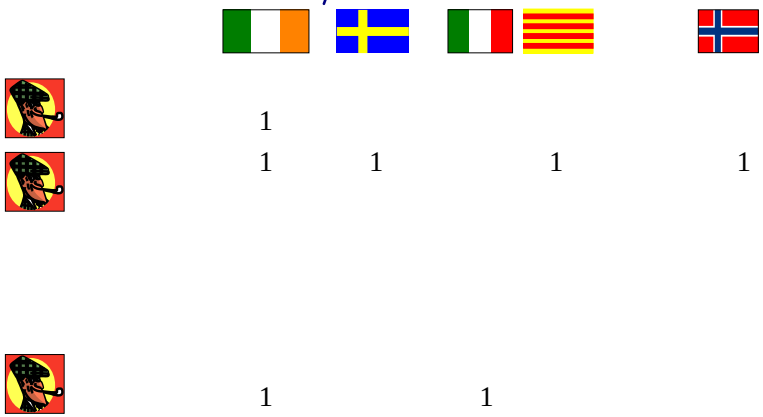
- Gilbert model $\mathcal{G}(n, p)$
  - n: number of nodes
  - p: each edge is chosen with probability p
- That is, $E = \{e_{ij}\}_{ij}$, $e_{ij} \in \{0, 1\}$ and $e_{ij} = 1$ with probability p

Noise addition: Graphs to add

Graphs. For bipartite graphs

- Gilbert model $\mathcal{G}(n, m, p)$
  - n, m: number of nodes each part U, V
  - p: each edge $(U - V)$ is chosen with probability p

Preferences/likes: travellers vs. countries; customers vs. products



Differential privacy

Definition. For $0 < p < 1/2$, we define the noise-graph protection mechanism as:

$$\mathcal{A}_{n,p}(G) = E(G \oplus g)$$

with $g \in \mathcal{G}(n,p)$ (Gilbert model)

Theorem. This mechanism provides $ln((1-p)/p)$-differential privacy

- This is for edge-differential privacy: Presence/absence of an edge does not make a difference: hiding individual edges

Differential privacy

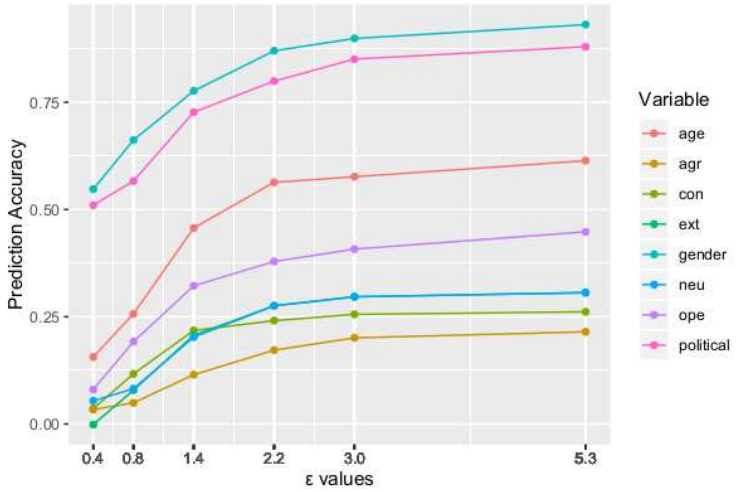
Example. Facebook likes data (after trimming, min 50 likes, 150 users/like)

(19,724 users, 8,523 likes, 3,817,840 user-like pairs)

- Analysis:

p	ϵ	$\lvert E(g)\rvert$	$\lvert E(G \oplus g)\rvert$
0.005	5.29	840,162	4,619,770
0.05	2.94	8,408,449	11,844,981
0.1	2.19	16,824,538	19,878,770
0.2	1.38	33,657,261	35,949,261
0.4	0.40	67,302,556	68,070,070

- Prediction accuracy for gender, age, political views, ...



Analysis of communities

Analysis of communities[5]

- Community detection using singular value decomposition + clustering

Approach:

- Use signless Laplacian matrix
$$|L| = D + A$$
where D: diagonal matrix with node degrees, A: adjacency matrix
- Matrix factorization of $|L|$ using SVD. Nodes as vectors in terms of orthogonal bases and singular values.
- Reduced dimensional approximation $|L|'$
- Similarity between pairs of vertices using dot products of vectors
- Clustering of vertices
(fuzzy clustering to permit multiple memberships to communities)

[5] VT, Graph addition: properties for its use for graph protection, ILAS 2020 (hold in Galway 2022 :))

Analysis of communities

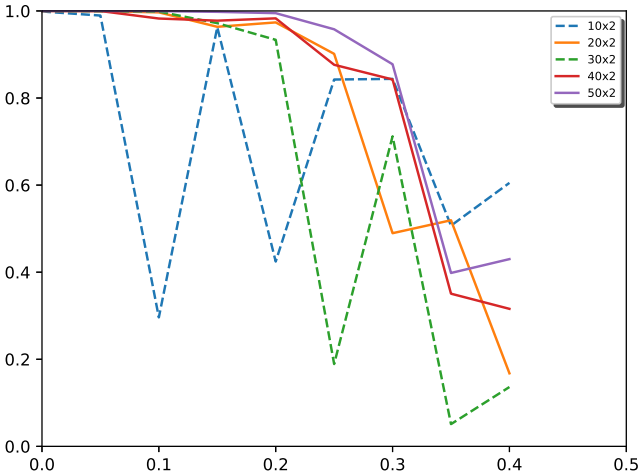
Example.

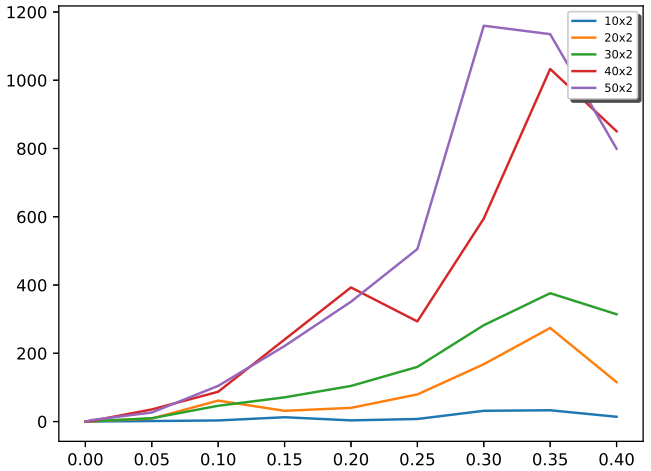
- Two communities. Gilbert model $G \sim \mathcal{G}(n, m, p_n, p_m, p_{nm})$
- Community detection for graph addition

$$G_p = G \oplus g_p$$

with $g_p \sim \mathcal{G}(n+m, p)$ and
$p \in \{0, 0.005, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4\}$

- Membership correlation between G and G_p





Extension to dynamic graphs

Analysis of communities

- Graph evolves with time. Snapshots of graphs.
- Edge-local differential privacy for dynamic graphs
  - $\mathcal{A}$ satisfies ε-edge local DP if for all nodes u, v, times stamps t and edge values i, j, k:

$$P(\mathbb{1}_{\mathcal{A}(uv(t))} = y | \mathbb{1}_{uv(t)} = x) \leq e^{\epsilon} P(\mathbb{1}_{\mathcal{A}(uv(t))}) = y | \mathbb{1}_{uv(t)} = x'),$$

- Parallel protection mechanism: $\mathcal{A}^{||}_{p_0,p_1}(G)$
  - $G = G_0, G_1, \ldots, G_T$ a dynamic graph, $\mathcal{A}_{p_0,p_1}$ a noise-graph mechanism, produce

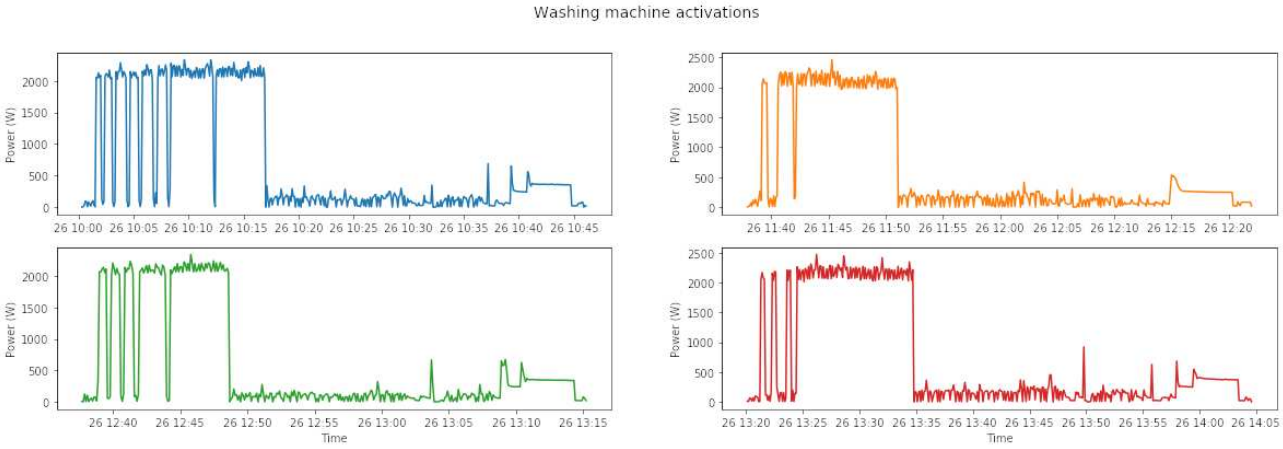
$$\tilde{G} = \tilde{G}_0, \tilde{G}_1, \ldots, \tilde{G}_T$$

with $\tilde{G}_i = \mathcal{A}_{p_0,p_1}(G_i)$ for $i = 0, \ldots, T$.

Smart grid

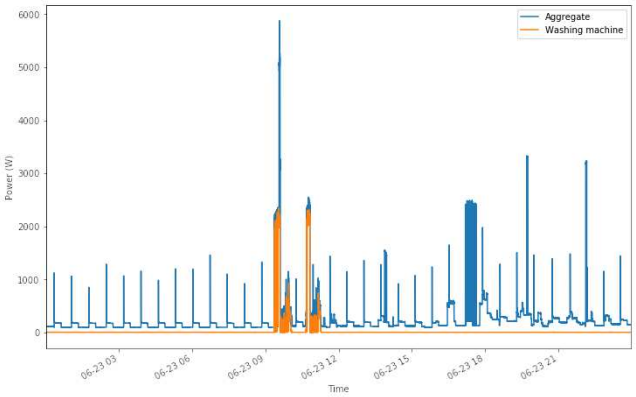
Temporal data: smart grid

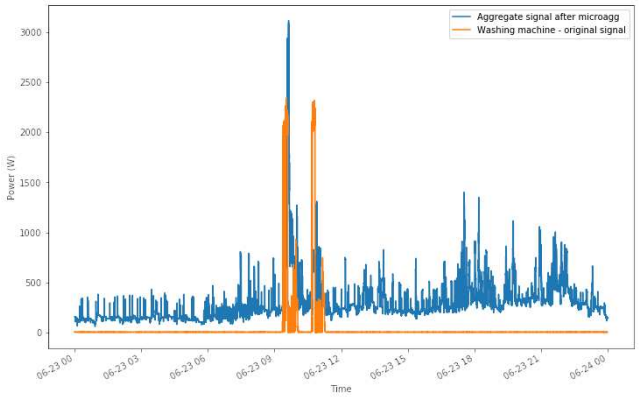
- Smart grid: electric grid data
  - Data from households
- Sensitive data:
  - consumer habits,
  - Non-intrusive load monitoring (NILM): deduce types of appliances from aggregated energy consumption.

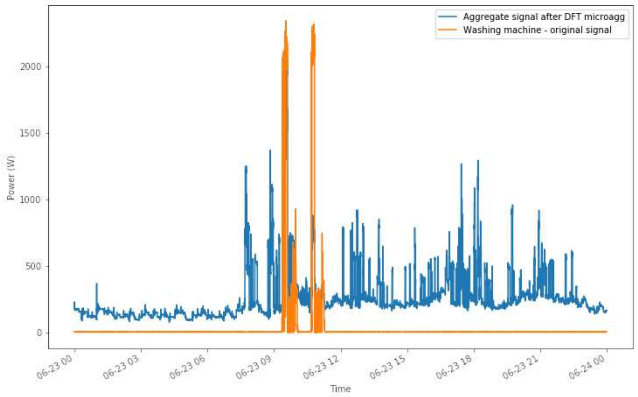


Temporal data: smart grid

- Our approach:
  - Data is centralized by the service provider
  - Data needs to be shared without disclosure
- Protection through microaggregation and DFT







Temporal data: smart grid

- Data utility based on data mining tasks[6]:
  - clustering: k-means
  - classification (type of consumer): kNN
  - forecasting: mean hourly load forecasting using SARIMAX model (seasonal ARIMA)
- Adversarial model:
  - Re-identification (based on record linkage)
  - Interval disclosure (is the masked value too similar?)
  - Non-intrusive load monitoring (NILM) detection.

[6] K. Adewole, V. Torra, DGTMicroagg: a dual-level anonymization algorithm for smart grid data, Int. J. of Inf. Systems 2022; K. Adewole, V. Torra, On the application of microaggregation and discrete Fourier transform for energy disaggregation risk reduction, submitted.

The case of federated Learning

Federated learning

- Motivation
  - Symbolic models (decision trees) vs. numerical models (deep learning)
  - Comparison of different privacy models: local and global
  - Who we trust? (privacy as a matter of trust)

Federated learning

- Privacy in federated learning and trust
  - Local privacy: The agent does not trust the system
    Local-DP / k-anonymity / privacy for re-identification

Federated learning

- Privacy in federated learning and trust
  - Local privacy: The agent does not trust the system
    Local-DP / k-anonymity / privacy for re-identification
  - Global privacy: Only globally we can really protect individuals
    (global) DP

Federated learning

- Privacy in federated learning and trust
  - Local privacy: The agent does not trust the system
    Local-DP / k-anonymity / privacy for re-identification
  - Global privacy: Only globally we can really protect individuals
    (global) DP
  - Infrastructure: No one trusts any one
    Secure multiparty computation

Federated learning

- Privacy in federated learning and trust
  - Local privacy: The agent does not trust the system
    Local-DP / k-anonymity / privacy for re-identification
  - Global privacy: Only globally we can really protect individuals
    (global) DP
  - Infrastructure: No one trusts any one
    Secure multiparty computation
  - Data in the cloud
    Homomorphic encryption

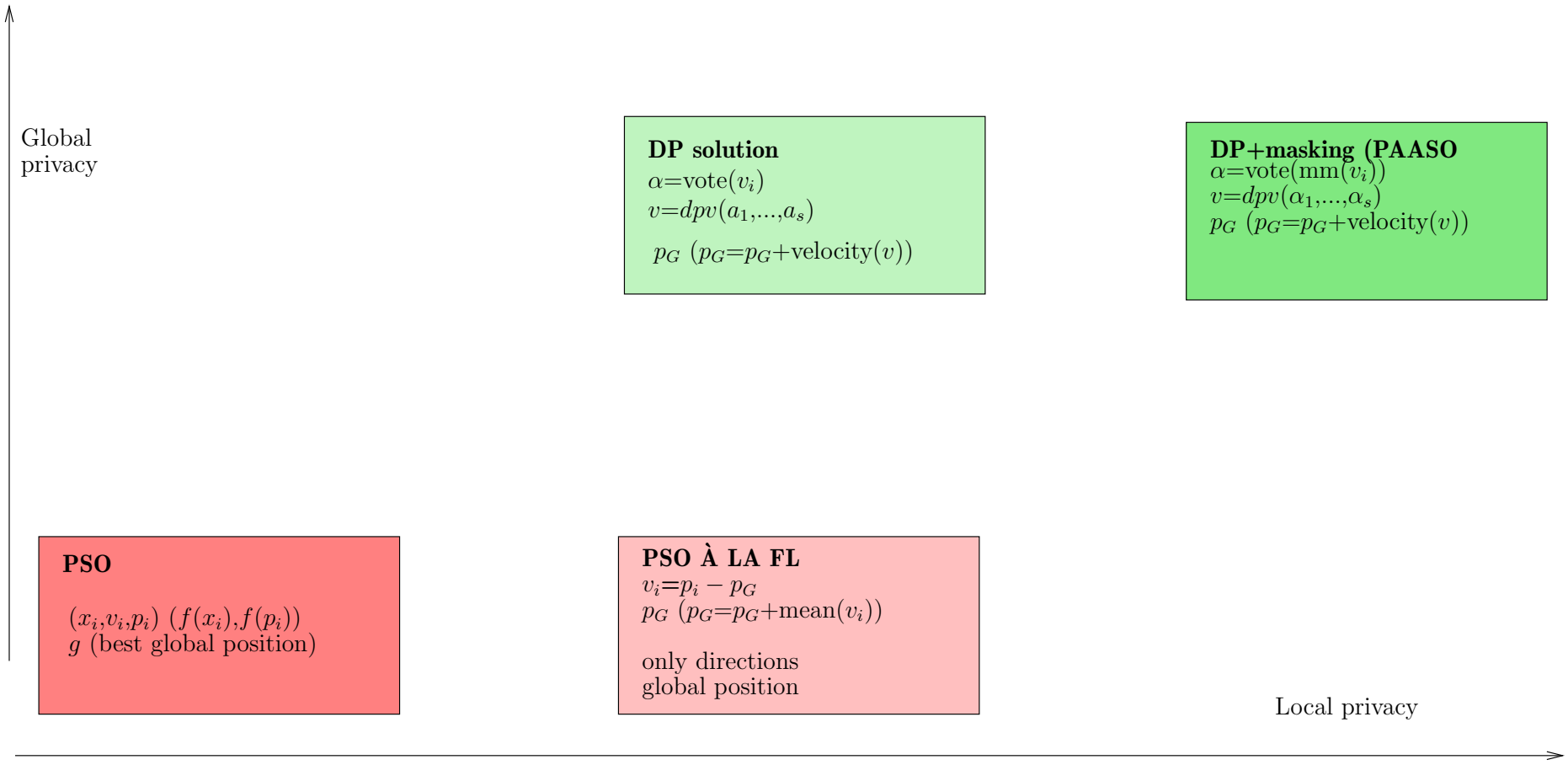
Federated learning

- Framework based on Particle Swarm Optimization (PSO)
  - Function to minimize
  - A set of particles (moving, position, direction) to find optimal
  - Privacy as a matter of trust
- Privacy at different levels
  - Symbolic vs. numerical PSO (less precision/voting/masking)
    - PSO is numerical, public (no-privacy): directions and positions
    - PSO à la FL (i.e., privacy):
      discrete directions (set of possible angles) – voting
  - Local vs. global privacy (individuals/clients vs. server)
    - Local: Masking vote before casting it (PRAM)
    - Global: Differentially private voting using DP-Random dictatorship[7]

[7] V. Torra, Random dictatorship for privacy-preserving social choice, Int. J. of Inf Sec, 19:5 (2020)

Federated learning

- PSO + FL = PAASO: Privacy-aware agent swarm optimization



Federated learning

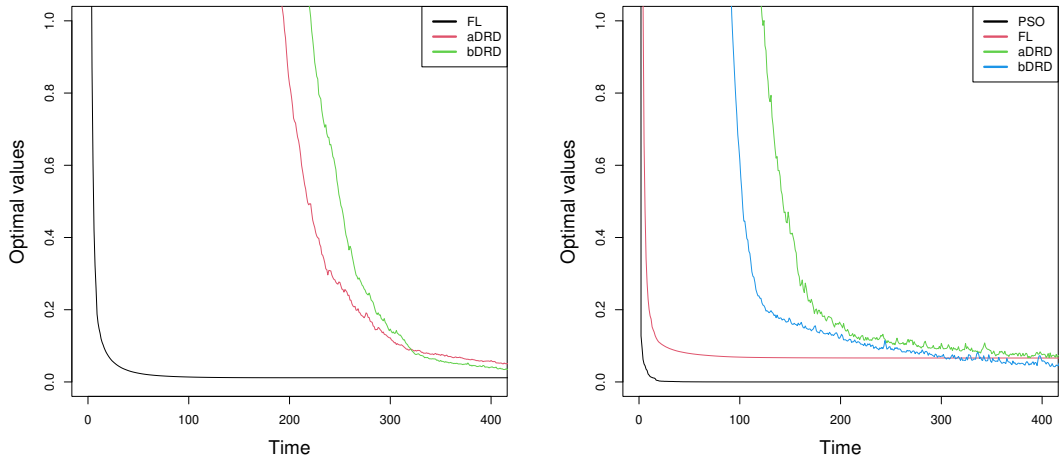
- General comments PAASO with 2D problems[8]
  - In general, privacy mechanisms do not avoid convergence. It is slower. (this can be of concern, of course, rounds=information)
  - In terms of convergence, PSO and FL are best.
  - Local protection (PRAM) does not have strong effect.
- On the parameters
  - Number of options in voting, low effect
  - Number of agents, key factor (50, 100, and 200)
  - Particular parameters depend on the problem + privacy strategy

[8] V Torra et al., PSO + FL = PAASO: particle swarm optimization + federated learning = privacy-aware agent swarm optimization. Int. J. Inf. Sec. (2022)

Federated learning

- An example:
  - Mean objective function for 20 executions for FL, aDRD, and bDRD. Function f_4, number of voting alternatives $k_\alpha = 8$, 50 agents, $\phi_p = \phi_g = 2.00$. $p_c = 1.0$.
  - (left) $\omega = 4.00$, $\omega_G = 0.005$; (right) $\omega = 0.005$, $\omega_G = 0.01$
  - Generalized Rosenbrock's function ($x_1, x_2 \in [-2.0, 2.0]$):

$$f_4(x_1, x_2) = 100 * (x_2 - x_1 * x_1)^2 + (x_1 - 1)^2$$



Centralized approach: Trusted third party

Trusted third party

Computation-driven approaches/multiple databases: centralized

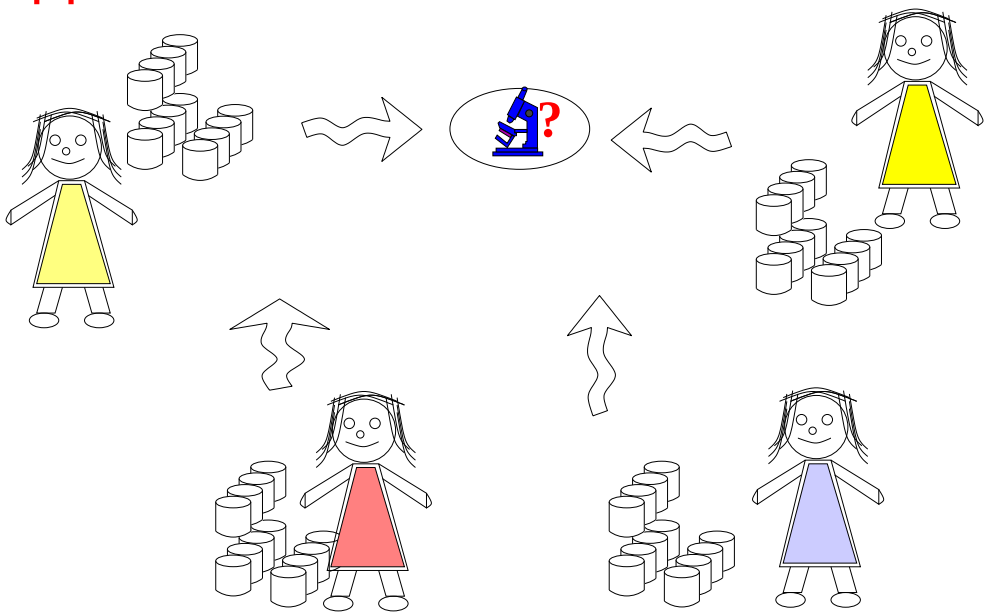
- **Example.** Parties $P_1, \ldots, P_n$ own databases $DB_1, \ldots, DB_n$. The parties want to compute a function, say f, of these databases (i.e., $f(DB_1, \ldots, DB_n)$) without revealing unnecessary information. In other words, after computing $f(DB_1, \ldots, DB_n)$ and delivering this result to all P_i, what P_i knows is nothing more than what can be deduced from his DB_i and the function f.
- So, the computation of f has not given P_i any extra knowledge.

Distributed approach: secure multiparty computation

Secure multiparty computation

Computation-driven approaches/multiple databases: distributed

- The centralized approach as a reference



Secure multiparty computation

Computation-driven approaches/multiple databases/distributed. Sum

- Compute the sum of salaries of 4 people: Aine, Brianna, Cathleen, and Deirdre.
  We denote these salaries by s_1, s_2, s_3, and s_4, respectively.
- Each person's salary is confidential and they do not want to share.
- Define a protocol to compute involving only the 4 people (no trusted third party).
- Assume that the sum lies in the range $[0, n]$.

□ Example with 4 people. Similar method applies with other number of people.

□ We use public-key cryptography. I.e., each party requires two separate keys: a private and a public one. This is also known as asymmetric cryptography.

Secure multiparty computation

Computation-driven approaches/multiple databases/distributed. Sum

- Aine adds a secret random number, say r (uniformly chosen in $[0, n]$) to her salary and sends it to Brianna encrypted with Brianna public key. Addition is modulo n. In this way, the outcome of $r + s_1 \ mod \ n$ will be a number uniformly distributed in $[0, n]$ and so Brianna will learn nothing about the actual value of s_1.

Secure multiparty computation

Computation-driven approaches/multiple databases/distributed. Sum

- Aine adds a secret random number, say r (uniformly chosen in $[0, n]$) to her salary and sends it to Brianna encrypted with Brianna public key. Addition is modulo n. In this way, the outcome of $r + s_1 \ mod \ n$ will be a number uniformly distributed in $[0, n]$ and so Brianna will learn nothing about the actual value of s_1.
- Brianna decrypts Aine's message with Brianna's private key, adds her salary (modulo n) and sends the result (i.e., $r + s_1 + s_2 \ mod \ n$) to Cathleen encrypted with Cathleen's public key.

Secure multiparty computation

Computation-driven approaches/multiple databases/distributed. Sum

- Aine adds a secret random number, say r (uniformly chosen in $[0, n]$) to her salary and sends it to Brianna encrypted with Brianna public key. Addition is modulo n. In this way, the outcome of $r + s_1 \ mod \ n$ will be a number uniformly distributed in $[0, n]$ and so Brianna will learn nothing about the actual value of s_1.
- Brianna decrypts Aine's message with Brianna's private key, adds her salary (modulo n) and sends the result (i.e., $r + s_1 + s_2 \ mod \ n$) to Cathleen encrypted with Cathleen's public key.
- Cathleen decrypts Brianna's message with Cathleen's private key, adds her salary (modulo n) and sends the result (i.e., $r + s_1 + s_2 + s_3 \ mod \ n$) to Deirdre encrypted with Deirdre's public key.

Secure multiparty computation

Computation-driven approaches/multiple databases/distributed. Sum

- Aine adds a secret random number, say r (uniformly chosen in $[0, n]$) to her salary and sends it to Brianna encrypted with Brianna public key. Addition is modulo n. In this way, the outcome of $r + s_1 \ mod \ n$ will be a number uniformly distributed in $[0, n]$ and so Brianna will learn nothing about the actual value of s_1.
- Brianna decrypts Aine's message with Brianna's private key, adds her salary (modulo n) and sends the result (i.e., $r + s_1 + s_2 \ mod \ n$) to Cathleen encrypted with Cathleen's public key.
- Cathleen decrypts Brianna's message with Cathleen's private key, adds her salary (modulo n) and sends the result (i.e., $r + s_1 + s_2 + s_3 \ mod \ n$) to Deirdre encrypted with Deirdre's public key.
- Deirdre decrypts Cathleen's message with Deirdre's private key, adds her salary (modulo n) and sends the result (i.e., $r + s_1 + s_2 + s_3 + s_4 \ mod \ n$) to Aine encrypted with Aine's public key.

Secure multiparty computation

Computation-driven approaches/multiple databases/distributed. Sum

- Aine adds a secret random number, say r (uniformly chosen in $[0, n]$) to her salary and sends it to Brianna encrypted with Brianna public key. Addition is modulo n. In this way, the outcome of $r + s_1 \ mod \ n$ will be a number uniformly distributed in $[0, n]$ and so Brianna will learn nothing about the actual value of s_1.
- Brianna decrypts Aine's message with Brianna's private key, adds her salary (modulo n) and sends the result (i.e., $r + s_1 + s_2 \ mod \ n$) to Cathleen encrypted with Cathleen's public key.
- Cathleen decrypts Brianna's message with Cathleen's private key, adds her salary (modulo n) and sends the result (i.e., $r + s_1 + s_2 + s_3 \ mod \ n$) to Deirdre encrypted with Deirdre's public key.
- Deirdre decrypts Cathleen's message with Deirdre's private key, adds her salary (modulo n) and sends the result (i.e., $r + s_1 + s_2 + s_3 + s_4 \ mod \ n$) to Aine encrypted with Aine's public key.
- Aine decrypts Deirdre's message with Aine's private key. She substracts (modulo n) the random number r added in the first step, obtaining in this way $s_1 + s_2 + s_3 + s_4$ (this will be in $[0, n]$).

Secure multiparty computation

Computation-driven approaches/multiple databases/distributed. Sum

- Aine adds a secret random number, say r (uniformly chosen in $[0, n]$) to her salary and sends it to Brianna encrypted with Brianna public key. Addition is modulo n. In this way, the outcome of $r + s_1 \ mod \ n$ will be a number uniformly distributed in $[0, n]$ and so Brianna will learn nothing about the actual value of s_1.
- Brianna decrypts Aine's message with Brianna's private key, adds her salary (modulo n) and sends the result (i.e., $r + s_1 + s_2 \ mod \ n$) to Cathleen encrypted with Cathleen's public key.
- Cathleen decrypts Brianna's message with Cathleen's private key, adds her salary (modulo n) and sends the result (i.e., $r + s_1 + s_2 + s_3 \ mod \ n$) to Deirdre encrypted with Deirdre's public key.
- Deirdre decrypts Cathleen's message with Deirdre's private key, adds her salary (modulo n) and sends the result (i.e., $r + s_1 + s_2 + s_3 + s_4 \ mod \ n$) to Aine encrypted with Aine's public key.
- Aine decrypts Deirdre's message with Aine's private key. She substracts (modulo n) the random number r added in the first step, obtaining in this way $s_1 + s_2 + s_3 + s_4$ (this will be in $[0, n]$).
- Aine announces the result to the participants.

Secure multiparty computation

Computation-driven approaches/multiple databases/distributed. Sum

- This protocol assumes that all of the participants are honest
- A participant can lie about her salary.
- Aine can announce a wrong addition.
- Participants can collude. E.g.,
  - Brianna and Deirdree can share their figures to find the salary of Cathleen

Secure multiparty computation

Computation-driven approaches/multiple databases/distributed. Sum

- Solving collusion.
  - Each salary is divided into shares.
  - The sum of each share is computed individually.
  - Different paths are used for different shares in a way that neighbors are different.
    To compute any s_i all neighbors of all paths are required.
  - Different number of shares imply different minimum coalition sizes for violating security

Secure multiparty computation

Computation-driven approaches/multiple databases/distributed. Sum

Important observation

- This method is compliant with the privacy model selected: Secure multiparty computation
- This method is not compliant with other privacy models: differential privacy

We can define appropriate methods that satisfy multiple privacy models

- E.g., method that computes differentially private secure sum

Secure multiparty computation

Computation-driven approaches/multiple databases/distributed. Sum

- We can also apply Shamir's secret sharing approach to this problem

Secure multiparty computation

Computation-driven approaches/multiple databases/distributed.

- Yao's millionaire problem. Alice and Bob want to know who is richer, but they do not want to tell the other how much money they have. This is the secure computation of $a > b$.
- Secure set union.
- Scalar product. Alice with vector x and Bob with vector y want to compute xy.

Secure multiparty computation

Computation-driven approaches/multiple databases/distributed.

- Dining Cryptographers Problem.
  - (Chaum, 1985) Three cryptographers are sitting down to dinner at their favorite three-star restaurant. Their waiter informs them that arrangements have been made with the maître d'hôtel for the bill to be paid anonymously. One of the cryptographers might be paying the dinner, or it might have been NSA (U.S. National Security Agency). The three cryptographers respect each other's right to make an anonymous payment, but they wonder if NSA is paying.
- This problem (and previous ones) can be seen from a user's privacy perspective (more particularly, about protecting the data of the user). I.e., the cryptographers does not want to share whether they paid or not.

Secure multiparty computation

Computation-driven approaches/multiple databases: distributed.

- Machine learning and data mining methods.
- Parties can be seen as sharing the schema of a database.
- Two types of problems usually considered.
  - Vertically partitioned data. Parties (data holders) have information on the same individuals but different attributes.
  - Horizontally partitioned data. Parties (data holders) have information on different individuals but on the same attributes (i.e., the share the database schema).

Secure multiparty computation

Computation-driven approaches/multiple databases: distributed

Privacy leakage for the distributed approach is usually analyzed considering two types of adversaries.

Secure multiparty computation

Computation-driven approaches/multiple databases: distributed

Privacy leakage for the distributed approach is usually analyzed considering two types of adversaries.

- **Semi-honest adversaries.** Data owners follow the cryptographic protocol but they analyse all the information they get during its execution to discover as much information as they can.
- **Malicious adversaries.** Data owners try to fool the protocol (e.g. aborting it or sending incorrect messages on purpose) so that they can infer confidential information.