INFO-H420
Management of Data Science and Business Workflows

Part II
Data Privacy

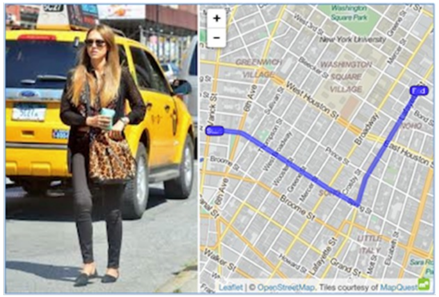
Dimitris SACHARIDIS

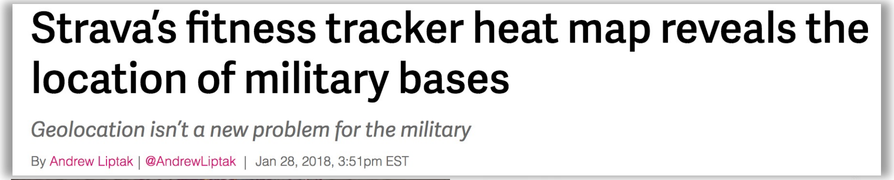
2023-2024

Data Release

Data Release Goes Bad

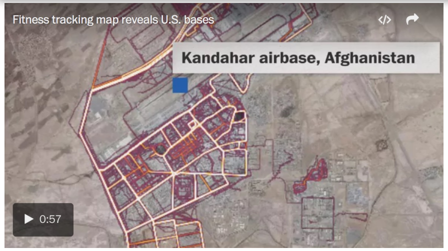
NYC Taxi & Limousine Commission


Jessica Alba (actor)


Strava's fitness tracker heat map reveals the location of military bases

Geolocation isn't a new problem for the military

By Andrew Liptak | @AndrewLiptak | Jan 28, 2018, 3:51pm EST


Fitness tracking map reveals U.S. bases
Kandahar airbase, Afghanistan
0:57

AOL

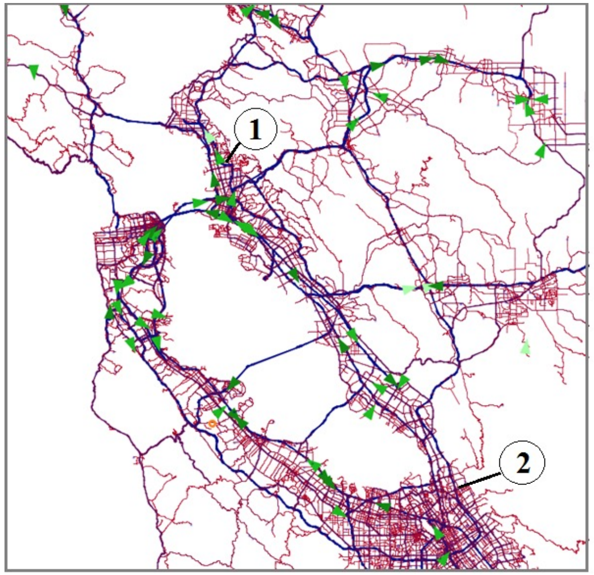
NETFLIX

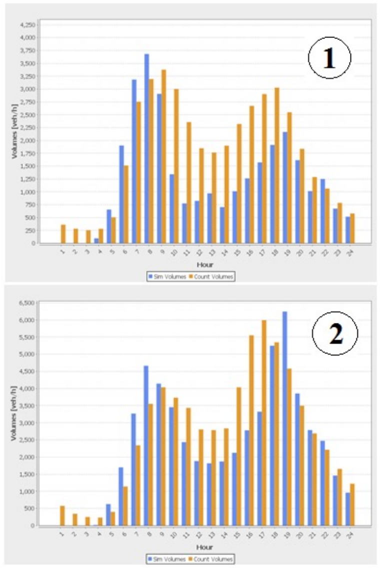
Commonwealth of Massachusetts Group Insurance Commission

Your Benefits Connection

We need to solve this data release problem ...

But Why Release Data?





- Urban mobility modeling at scale
  - Smart traffic design is critical, congestion is getting worse

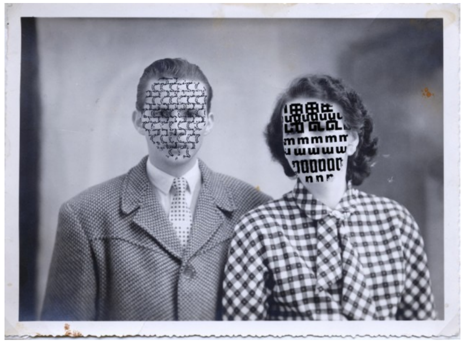
ULB

Big Data is Expected to Solve Big Problems

- The big data concept has taken off
  - Many organizations have adopted it
  - Entered the public vocabulary

- But this data is mostly about individuals
  - Individuals want privacy for their data

- How can researchers work on sensitive data?
  - The easy answer: anonymize it and share
  - The problem: we don't know how to do this



Data Release

- Several agencies, institutions, bureaus, organizations make (sensitive) data involving people publicly available
  - termed microdata (vs. aggregated macrodata) used for analysis
  - often required and imposed by law

- To protect privacy microdata are anonymized
  - explicit identifiers (SSN, name, phone #) are removed

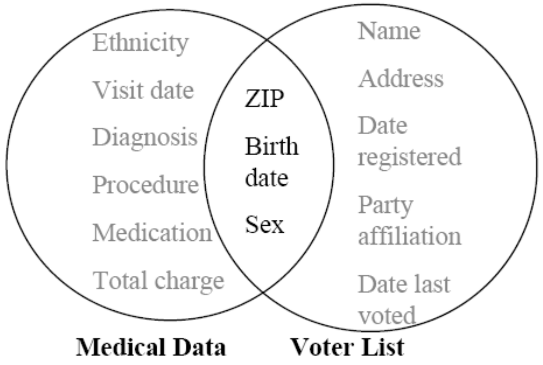
- is this sufficient for preserving privacy?
- no! susceptible to link attacks
  - publicly available databases (voter lists, city directories) can reveal the "hidden" identity

ULB

Link attack example

- Sweeney [S02a] managed to re-identify the medical record of the governor of Massachussetts
  - MA collects and publishes anonymized medical data for state employees (microdata) left circle
  - voter registration list of MA (publicly available data) right circle

- looking for governor's record
- join the tables:
  - 6 people had his birth date
  - 3 were men
  - 1 in his zipcode



- regarding the US 1990 census data
  - 87% of the population are unique based on (zipcode, gender, dob)

[S02a] Sweeney, L. *k*-Anonymity: A Model for Protecting Privacy. International Journal on Uncertainty, Fuzziness and Knowledge-based Systems, 2002.

ULB

Privacy in Data Publishing

- the role of attributes in published data
  - explicit identifiers are removed
  - quasi identifiers (QI) can be used to re-identify individuals
  - sensitive attributes (may not exist!) carry sensitive information

identifier	quasi identifiers			sensitive
Name	**Birthdate**	**Sex**	**Zipcode**	**Disease**
Andre	21/1/79	male	53715	Flu
Beth	10/1/81	female	55410	Hepatitis
Carol	1/10/44	female	90210	Bronchitis
Dan	21/2/84	male	02174	Sprained Ankle
Ellen	19/4/72	female	02237	AIDS

goal of privacy preservation (rough definition)
de-associate individuals from sensitive information

k-anonymity

- preserve privacy via k-anonymity, proposed by Sweeney and Samarati [S01, S02a, S02b]
- k-anonymity: intuitively, hide each individual among k-1 others
  - each QI set of values should appear at least k times in the released microdata
  - linking cannot be performed with confidence > 1/k
  - sensitive attributes are not considered

- how to achieve this?
  - generalization and suppression
  - value perturbation is not considered (we should remain *truthful* to original values)

- privacy vs utility tradeoff
  - do not anonymize more than necessary

[S01] Samarati, P. Protecting Respondents' Identities in Microdata Release. IEEE TKDE, 13(6):1010-1027, 2001.
[S02a] Sweeney, L. *k*-Anonymity: A Model for Protecting Privacy. International Journal on Uncertainty, Fuzziness and Knowledge-based Systems, 2002.
[S02b] Sweeney, L. *k*-Anonymity: Achieving *k*-Anonymity Privacy Protection using Generalization and Suppresion. International Journal on Uncertainty, Fuzziness and Knowledge-based Systems, 2002.

k-anonymity example

tools for anonymization

- generalization
  - publish more general values, i.e., given a domain hierarchy, roll-up
- suppression
  - remove tuples, i.e., do not publish outliers
  - often the number of suppressed tuples is bounded

original data

Birthdate	**Sex**	**Zipcode**
21/1/79	male	53715
10/1/79	female	55410
1/10/44	female	90210
21/2/83	male	02274
19/4/82	male	02237

2-anonymous data

	Birthdate	**Sex**	**Zipcode**
group 1	*/1/79	person	5****
	*/1/79	person	5****
suppressed	1/10/44	female	90210
group 2	*/*/8*	male	022**
	//8*	male	022**

k-anonymity problems

- homogeneity attack: in the last group everyone has cancer
- background knowledge: in the first group, Japanese have low chance of heart disease
- we need to consider the sensitive values

Original data

id	Zipcode	Sex	National.	Disease
1	13053	28	Russian	Heart Disease
2	13068	29	American	Heart Disease
3	13068	21	Japanese	Viral Infection
4	13053	23	American	Viral Infection
5	14853	50	Indian	Cancer
6	14853	55	Russian	Heart Disease
7	14850	47	American	Viral Infection
8	14850	49	American	Viral Infection
9	13053	31	American	Cancer
10	13053	37	Indian	Cancer
11	13068	36	Japanese	Cancer
12	13068	35	American	Cancer

4-anonymous data

id	Zipcode	Sex	National.	Disease
1	130**	<30	*	Heart Disease
2	130**	<30	*	Heart Disease
3	130**	<30	*	Viral Infection
4	130**	<30	*	Viral Infection
5	1485*	≥40	*	Cancer
6	1485*	≥40	*	Heart Disease
7	1485*	≥40	*	Viral Infection
8	1485*	≥40	*	Viral Infection
9	130**	3*	*	Cancer
10	130**	3*	*	Cancer
11	130**	3*	*	Cancer
12	130**	3*	*	Cancer

l-diversity (ell diversity)

- make sure each group contains well represented sensitive values
  - protect from homogeneity attacks
  - protect from background knowledge

> **l-diversity** (simplified definition)
> a group is l-diverse if the most frequent sensitive value appears at most 1/l times in group

- l-diversity definition has monotonicity properties similar to k-anonymity
  - Easy to adapt existing algorithms to count frequencies of sensitive values

Differential Privacy

Differential Privacy

- Differential privacy protects the privacy of individuals by **adding noise** or perturbation in a **controlled** way
- Goal: to make it **difficult to determine the presence or absence of any particular individual**

- It is a mathematical framework that provides a way to **measure** the **privacy loss** of a given dataset
- It allows for the creation of **algorithms** that can **analyze datasets** while still preserving the privacy of individuals within the dataset
- It provides a way to **balance** the **trade-off** between **utility** and **privacy**

Differential Privacy

A randomized algorithm K satisfies ε-differential privacy if:
Given two data sets that differ by one individual, D and D', any property S, and $\varepsilon > 0$:

$$\Pr[K(D) \in S] \leq e^{\varepsilon} \Pr[K(D') \in S]$$

- Can achieve ε-DP for counts by adding a random noise value
- Uncertainty due to noise → plausible deniability
  - Hides whether someone is present in the data

Achieving ε-DP

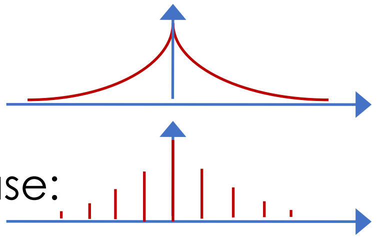
(Global) Sensitivity of publishing:

$s = \max_{D,D'} |F(D) - F(D')|$, D, D' differ by 1 individual

E.g., count individuals satisfying property P: $s = 1$

For every value that is output:

- Add Laplacian noise $Lap(s/\varepsilon)$:
- Or Geometric noise for discrete case:



Simple rules for composition of differentially private outputs:
Given output O_1 that is ε_1-DP and O_2 that is ε_2-DP

- (Sequential composition) If inputs overlap, result is $(\varepsilon_1 + \varepsilon_2)$-DP
- (Parallel composition) If inputs disjoint, result is $\max(\varepsilon_1, \varepsilon_2)$-DP

Trying to Reduce Trust

- Most work on differential privacy assumes a trusted party
  - Data aggregator (e.g., organizations) that sees the true, raw data
  - Can compute exact query answers, then perturb for privacy

- A reasonable question: can we reduce the amount of trust?
  - Can we remove the trusted party from the equation?
  - Users produce locally private output, aggregate to answer queries

- One approach is to use secure multiparty computation or homomorphic encryption
  - Merge encrypted data, and add noise for privacy inside encryption
  - Complex to get right, and very high computational overhead

Local Differential Privacy

- What about having each user run a DP algorithm on their data?
  - Then combine all the results to get a final answer
- On first glance, this idea seems crazy
  - Each user adds noise to mask their own input
  - So surely the noise will always overwhelm the signal?
- But … noise can cancel out or be subtracted out
  - We end up with the true answer, plus noise which can be smaller
  - However, noise is still larger than in the centralized case

Local Differential Privacy: Example

- Each of N users has 0/1 value, estimate total population sum
  - Each user adds independent Laplace noise: mean 0, variance $2/\varepsilon^2$

- Adding user results: true answer + sum of N Laplace distributions
  - Error is random variable, with mean 0, variance $2N/\varepsilon^2$
  - Confidence bounds: ~95% chance of being within 2σ of the mean
  - So, error looks like $\sqrt{N}/\varepsilon$, but true value may be proportional to N

- Numeric example: suppose true answer is $N/2$, $\varepsilon = 1$, $N = 1M$
  - We see $500K \pm 2800$: about 1% uncertainty
  - Error in centralized case would be close to 1 (0.001%)

Local Differential Privacy

- We can achieve LDP, and obtain reasonable accuracy (for large N)
  - The error typically scales with $\sqrt{N}$
- Generic approach: apply centralized DP algorithm to local data
  - But error might still be quite large
  - Unclear how to merge private outputs (e.g., private clustering)
- So, we seek to design new LDP algorithms
  - Maximize the accuracy of the results
  - Minimize the costs to the users (space, time, communication)
  - Ensure that there is an accurate algorithm for aggregation

Privacy with a coin toss: Randomized Response

- Each user has a single bit of private information
  - Encoding e.g. political/sexual/religious preference, illness, etc.

- Randomize Response (RR): toss an unbiased coin [Warner 65]
  - If Heads (probability $p = ½$), report the true answer
  - Else, toss unbiased coin again: if Heads, report true answer, else lie

- Collect responses from N users, subtract noise
  - Error in the estimate is proportional to $1/\sqrt{N}$
  - Simple LDP algorithm with parameter $\varepsilon=\ln((¾)/(¼)) = \ln(3)$
  - Generalization: allow biased coins ($p \neq ½$)

Privacy in practice

- Differential privacy based on coin tossing is widely deployed!
  - In Google Chrome browser, to collect browsing statistics
  - In Apple iOS and MacOS, to collect typing statistics
  - In Microsoft Windows to collect telemetry data over time
  - This yields deployments of over 100 million users each

- All deployments are based on RR, but extend it substantially
  - To handle the large space of possible values a user might have

- Local Differential Privacy is state of the now
  - Randomized response invented in 1965: six decades ago!

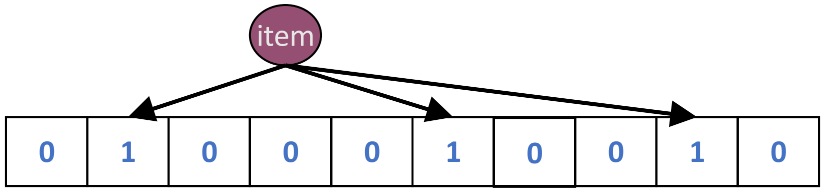
Google's RAPPOR

- Each user has one value out of a very large set of possibilities
  - E.g., their favourite URL, www.nytimes.com

- Attempt 1: run RR for all values: google.com? Bing.com? …
  - User has sparse binary vector with one 1: run RR on every bit
  - Satisfies 2ε-LDP: [recall ε=ln(3)] change user's choice, 2 bits change: 1 → 0, 0 → 1
  - Slow: sends 1 bit for every possible index in the vector
  - Limited: can't handle new possibilities being added

- Try to do better by reducing domain size through hashing

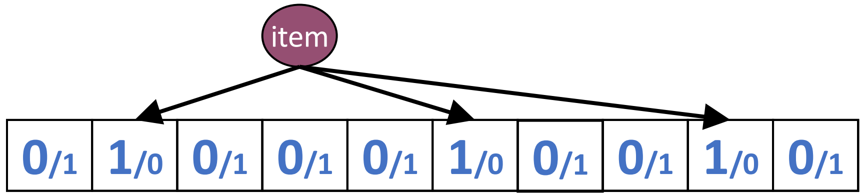
Bloom Filters



- Bloom filters [B70] compactly encode set membership
  - k hash functions map items to m-bit vector k times
  - Update: Set all k entries to **1** to indicate item is present
  - Query: Can lookup items, store set of size n in O(n) bits
  - Analysis: choose k and m to obtain small false positive probability

- Can be merged by OR-ing vectors (of same size)
  - Duplicate insertions do not change Bloom filters

Counting Bloom Filters & Randomized Response



- Idea: apply Randomized Response to the bits in a Bloom filter
  - Not too many bits in the filter compared to all possibilities
- Each user maps their input to at most k bits in the filter
  - Privacy guarantee with parameter 2kε
- Combine all user reports and observe how often each bit is set

Decoding Noisy Counting Bloom Filters

- We have a noisy Bloom filter, where each bit has a probability

- To estimate the frequency of a particular value:
  - Look up its bit locations in the Bloom filter
  - Compute the unbiased estimate of the probability each is 1
  - Take the minimum of these estimates as the frequency
  - More advanced decoding heuristics to decode all at once

- How to find frequent strings without knowing them in advance?
  - Subsequent work: build up frequent strings character by character

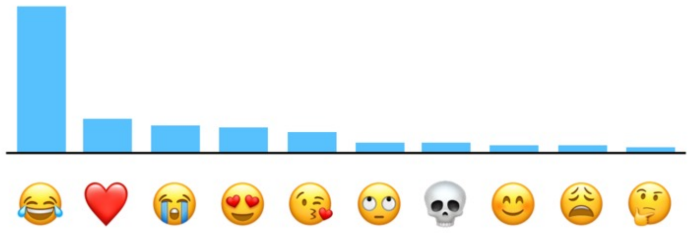
RAPPOR in practice

- The RAPPOR approach is implemented in the Chrome browser
  - Collects data from opt-in users, tens of millions per day
  - Open-source implementation available

- Tracks settings in the browser, e.g., home page, search engine
  - Many users unexpectedly change home page → possible malware

- Typical configuration:
  - 128 bit Bloom filter, 2 hash functions, privacy parameter ~0.5
  - Needs about 10K reports to identify a value with confidence

Apple's Differential Privacy in Practice



The Count Mean Sketch technique allows Apple to determine the most popular emoji to help design better ways to find and use our favorite emoji. The top emoji for US English speakers contained some surprising favorites.

- Apple uses their system to collect data from iOS and MacOS users
  - Popular emojis: (heart) (laugh) (smile) (crying) (sadface)
  - "New" words: bruh, hun, bae, tryna, despacito, mayweather
  - Which websites to mute, which to autoplay audio on!

- Deployment settings: w=1000, d=1000, ε=2-8 (some criticism)

Acknowledgements

- DP content from "*Privacy at Scale: Local Differential Privacy in Practice*" Tutorial in SIGMOD 2018 by Graham Cormode, Somesh Jha, Tejas Kulkarni, Ninghui Li, Divesh Srivastava, and Tianhao Wang