Distributed Private Data Collection at Scale

Graham Cormode

g.cormode@warwick.ac.uk

Tejas Kulkarni (Warwick)

Divesh Srivastava (AT&T)

THE UNIVERSITY OF WARWICK

Big data, big problem?

- The big data meme has taken root
  - Organizations jumped on the bandwagon
  - 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 Horror Stories

Commonwealth of Massachusetts
Group Insurance Commission
Your Benefits Connection

NETFLIX

AOL

NYC Taxi & Limousine Commission

We need to solve this data release problem...

THE UNIVERSITY OF WARWICK

Privacy is not Security

- Security is binary: allow access to data iff you have the key
  - Encryption is robust, reliable and widely deployed
- Private data release comes in many shades:
  reveal some information, disallow unintended uses
  - Hard to control what may be inferred
  - Possible to combine with other data sources to breach privacy
  - Privacy technology is still maturing
- Goals for data release:
  - Enable appropriate use of data while protecting data subjects
  - Keep CEO and CTO off front page of newspapers
  - Simplify the process as much as possible: 1-click privacy?

Differential Privacy (Dwork et al 06)

A randomized algorithm K satisfies ε-differential privacy if:

Given two data sets that differ by one individual, D and D', and any property S:

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

- Can achieve differential privacy for counts by adding a random noise value
- Uncertainty due to noise "hides" whether someone is present in the data

Privacy with a coin toss

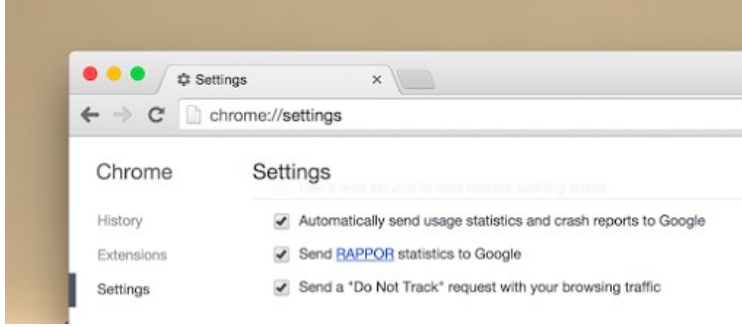
Perhaps the simplest possible DP algorithm

- Each user has a single private bit of information
  - Encoding e.g. political/sexual/religious preference, illness, etc.
- Toss a (biased) coin
  - With probability $p > ½$, report the true answer
  - With probability $1-p$, lie
- Collect the responses from a large number N of users
  - Can 'unbias' the estimate (if we know p) of the population fraction
  - The error in the estimate is proportional to $1/\sqrt{N}$
- Gives differential privacy with parameter $\ln (p/(1-p))$
  - Works well in theory, but would anyone ever use this?

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
  - This yields deployments of over 100 million users
- The model where users apply differential privately and then aggregated is known as "Local Differential Privacy"
  - The alternative is to give data to a third party to aggregate
  - The coin tossing method is known as 'randomized response'
- Local Differential privacy is state of the art in 2018: Randomized response invented in 1965: five decade lead time!

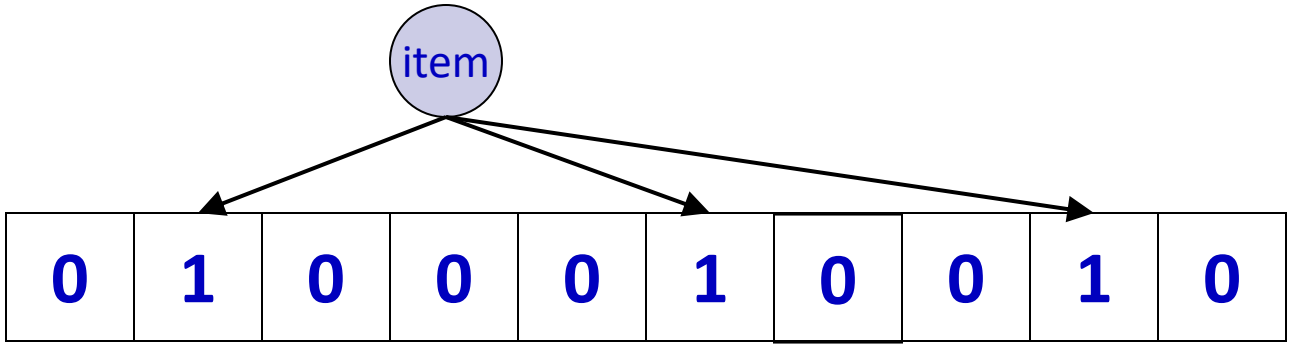
RAPPOR: Bits with a twist



- Each user has one value out of a very large set of possibilities
  - E.g. their favourite URL, www.bbc.co.uk
- First attempt: run randomized response for all possible values
  - Do you have google.com? Nytimes.com? Bing.com? Bbc.co.uk?...
- Meets required privacy guarantees with parameter $2 \ln(p/(1-p))$
  - If we change a user's choice, then at most two bits change: a 1 goes to 0 and a 0 goes to 1
- Slow: sends 1 bit for every possible choice
  - And limited: can't handle new options being added
- Try to do better by reducing domain size through hashing

Bloom Filters

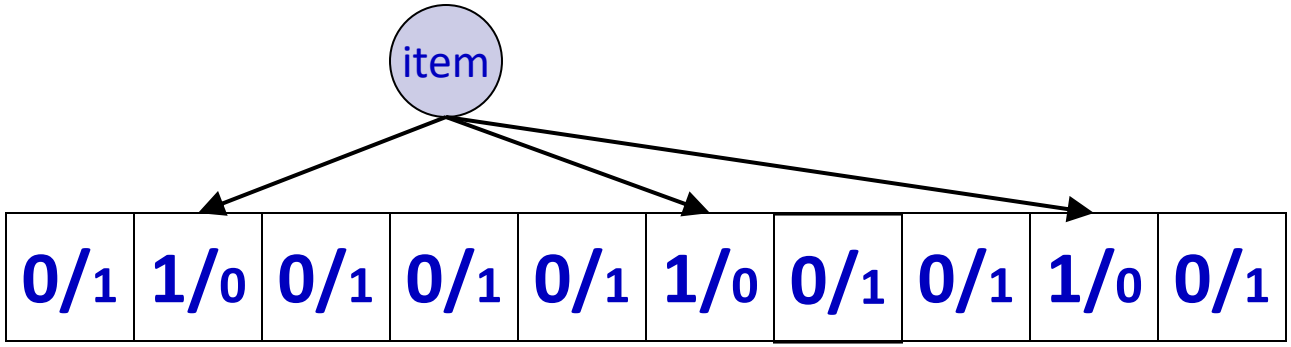
- Bloom filters [Bloom 1970] compactly encode set membership
  - E.g. store a list of many long URLs compactly
  - k hash functions map items to m-bit vector k times
  - Update: Set all k entries to **1** to indicate item is present
  - Query: Can lookup items, store set of size n in O(n) bits
    - Analysis: choose k and size m to obtain small false positive prob



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

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
  - New choices can be counted (by hashing their identities)
- Privacy guarantee with parameter k ln (p/(1-p))
  - Combine all user reports and observe how often each bit is set



Decoding noisy Bloom filters

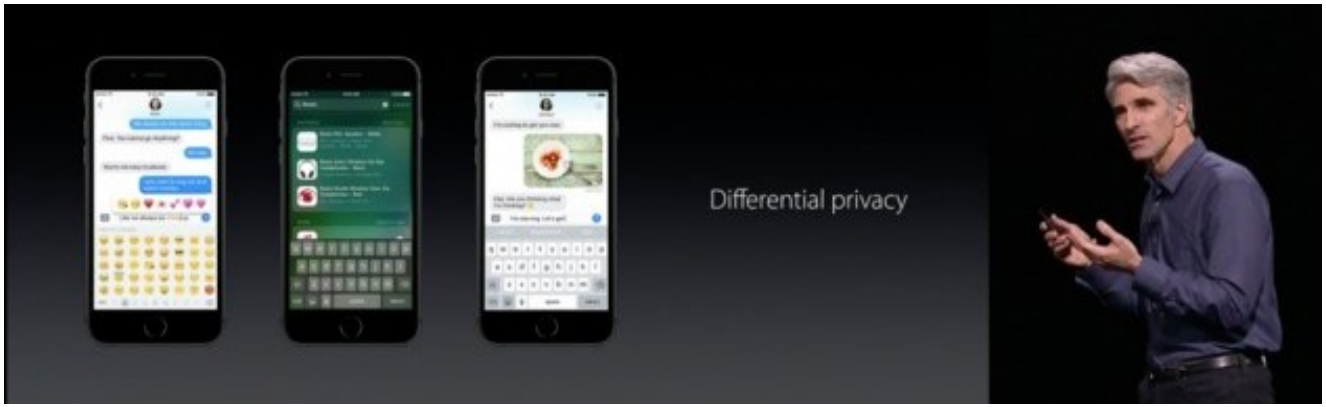
- We obtain a Bloom filter, where each bit is now a probability
- To estimate the frequency of a particular value:
  - Look up its bit locations in the Bloom filter
  - Compute the unbiased estimate of the probability each is 1
  - Take the minimum of these estimates as the frequency
- More advanced 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)
  - Identify if many users unexpectedly change their home page (indicative of malware)
- Typical configuration:
  - 128 bit Bloom filter, 2 hash functions, privacy parameter ~0.5
  - Needs about 10K reports to identify a value with confidence

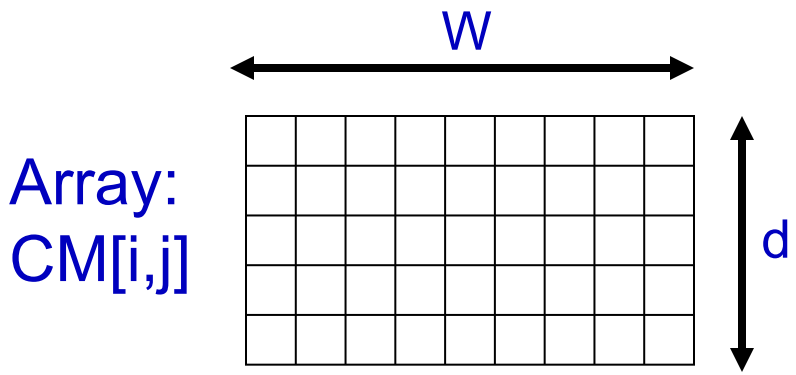
Apple: sketches and transforms

- Similar problem to Rappor:
  want to count frequencies of many possible items
  - For simplicity, assume each user holds a single item
  - Want to reduce the burden of collection:
    can we further reduce the size of the summary?
- Instead of Bloom Filter, make use of sketches
  - Similar idea, but better suited to capturing frequencies

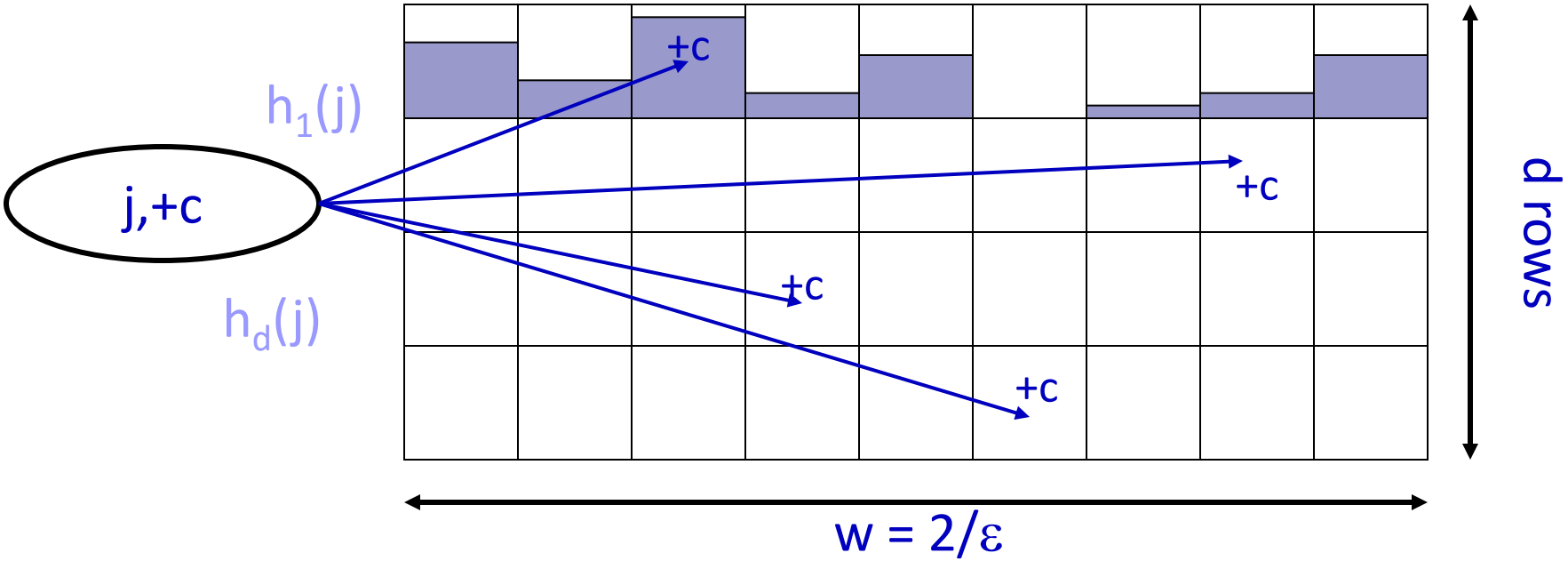


Count-Min Sketch

- Count Min sketch [C, Muthukrishnan 04] encodes item counts
  - Allows estimation of frequencies (e.g. for selectivity estimation)
  - Some similarities in appearance to Bloom filters
- Model input data as a vector x of dimension U
  - Create a small summary as an array of w × d in size
  - Use d hash function to map vector entries to [1..w]



Count-Min Sketch Structure



- Update: each entry in vector x is mapped to one bucket per row.
- Merge two sketches by entry-wise summation
- Query: estimate x[j] by taking $\min_k CM[k,h_k(j)]$
  - Guarantees error less than $\varepsilon\|x\|_1$ in size $O(1/\varepsilon)$
  - Probability of more error reduced by adding more rows

Count-Min Sketch + Randomized Response

- Each user encodes their (unit) input with a Count-Min sketch
  - Then applies randomized response to each entry
- Aggregator adds up all received sketches, unbiases the entries
- Take an unbiased estimate from the sketch based on mean
  - More robust than taking min when there is random noise
- Can bound the accuracy in the estimate via variance computation
  - Error is a random variable with variance proportional to $\|x\|_2^2/(wdn)$
  - I.e. (absolute) error decreases proportional to $1/\sqrt{n}$, $1/\sqrt{\text{sketch size}}$
- Bigger sketch → more accuracy
  - But we want smaller communication?

One weird trick: Hadamard transform

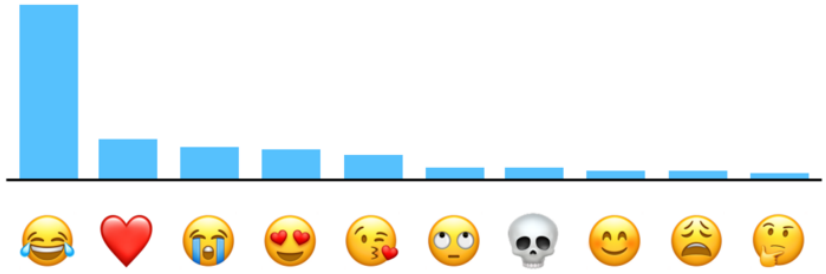
- The distribution of interest could be sparse and spiky
  - This is preserved under sketching
  - If we don't report the whole sketch, we might lose information
- Idea: transform the data to 'spread out'
  - Hadmard transform is a discrete Fourier
  - We will transform the sketched data

$$\begin{bmatrix} H^* & H^* \\ H^* & -H^* \end{bmatrix} = \begin{bmatrix} -1 & 1 & 1 & 1 & -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 & 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 & 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 & 1 & 1 & 1 & -1 \\ -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 \\ 1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 \\ 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 \\ 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 \end{bmatrix}.$$

- Aggregator reconstructs the transforme
  - Can invert the transform to get the sketch back
- Now the user just samples one entry in the transformed sketch
  - No danger of missing the important information – it's everywhere
  - Variance is essentially unchanged from previous case
- User only has to send one bit of information

Apple's Differential Privacy in Practice

- Apple use their system to collect data from iOS and OSX users
  - Popular emjois: (heart) (laugh) (smile) (crying) (sadface)
  - "New" words: bruh, hun, bae, tryna, despacito, mayweather
  - Which websites to mute, which to autoplay audio on!
  - Which websites use the most energy to render
- Deployment settings:
  - Sketch size w=1000, d=1000
  - Number of users not stated
  - Privacy parameter 2-8 (some criticism of this)



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.

Going beyond counts of data

- Simple frequencies can tell you a lot, but can we do more?
- Our recent work: materializing marginal distributions
  - Each user has d bits of data (encoding sensitive data)
  - We are interested in the distribution of combinations of attributes

	Gender	Obese	High BP	Smoke	Disease
Alice	1	0	0	1	0
Bob	0	1	0	1	1
…					
Zayn	0	0	1	0	0

Gender/Obese	0	1
0	0.28	0.22
1	0.29	0.21

Disease/Smoke	0	1
0	0.55	0.15
1	0.10	0.20

Nail, meet hammer

- Could apply Randomized Reponse to each entry of each marginal
  - To give an overall guarantee of privacy, need to change p
  - The more bits released by a user, the closer p gets to ½ (noise)
- Need to design algorithms that minimize information per user
- First observation: the sampling trick
  - If we release n bits of information per user, the error is $n/\sqrt{N}$
  - If we sample 1 out of n bits, the error is $\sqrt{(n/N)}$
  - Quadratically better to sample than to share!

THE UNIVERSITY OF WARWICK

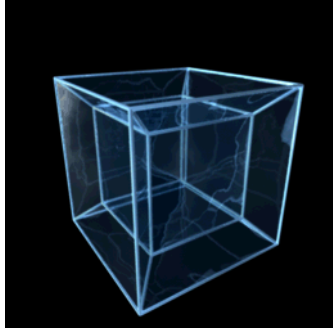
What to materialize?

Different approaches based on how information is revealed

1. We could reveal information about all marginals of size k
   - There are (d choose k) such marginals, of size 2^k each
2. Or we could reveal information about the full distribution
   - There are 2^d entries in the d-dimensional distribution
   - Then aggregate results here (obtaining additional error)

- Still using randomized response on each entry
  - Approach 1 (marginals): cost proportional to $2^{3k/2}\, d^{k/2}/\sqrt{N}$
  - Approach 2 (full): cost proportional to $2^{(d+k)/2}/\sqrt{N}$
- If k is small (say, 2), and d is large (say 10s), Approach 1 is better
  - But there's another approach to try…

Hadamard transform (again)



Instead of materializing the data, we can transform it

- The Hadamard transform is the
  Fourier transform for the binary
  - Very simple in practice
- Property 1: only (d choose k) co
  are needed to build any k-way m
  - Reduces the amount of information to release
- Property 2: Hadamard transform is a linear transform
  - Can estimate global coefficients by sampling and averaging
- Yields error proportional to $2^{k/2}d^{k/2}/\sqrt{N}$
  - Better than both previous methods (in theory)

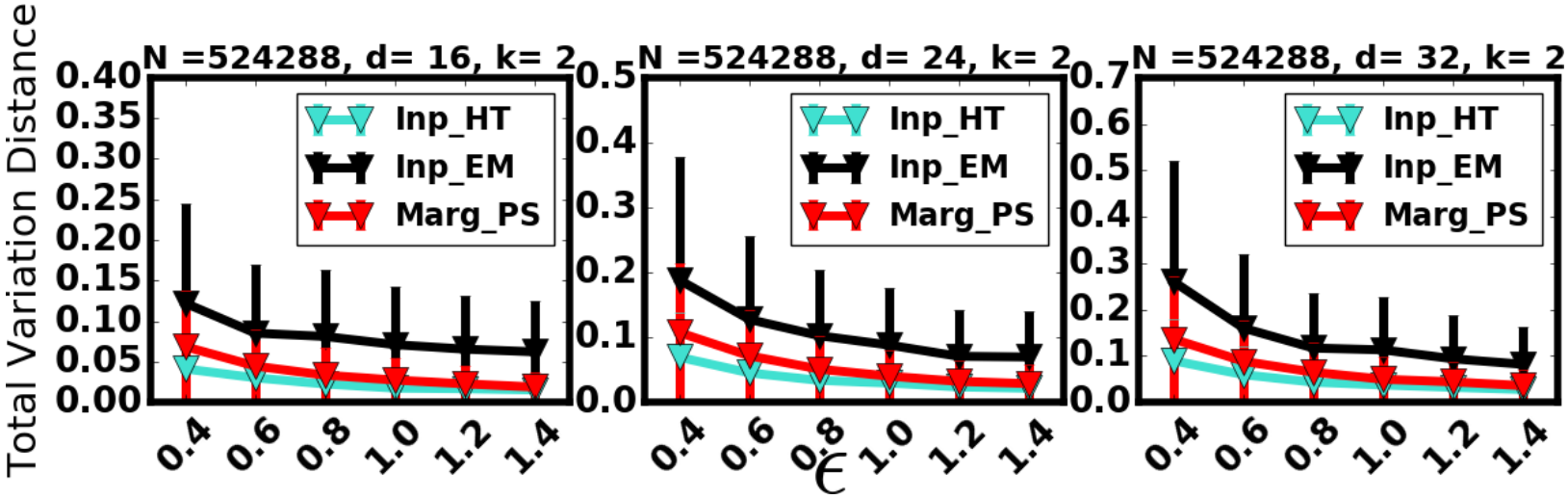
$$\begin{bmatrix} H^* & H^* \\ H^* & -H^* \end{bmatrix} = \begin{bmatrix} -1 & 1 & 1 & 1 & -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 & 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 & 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 & 1 & 1 & 1 & -1 \\ -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 \\ 1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 \\ 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 \\ 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 \end{bmatrix}.$$

Outline of error bounds

How to prove these error bounds?

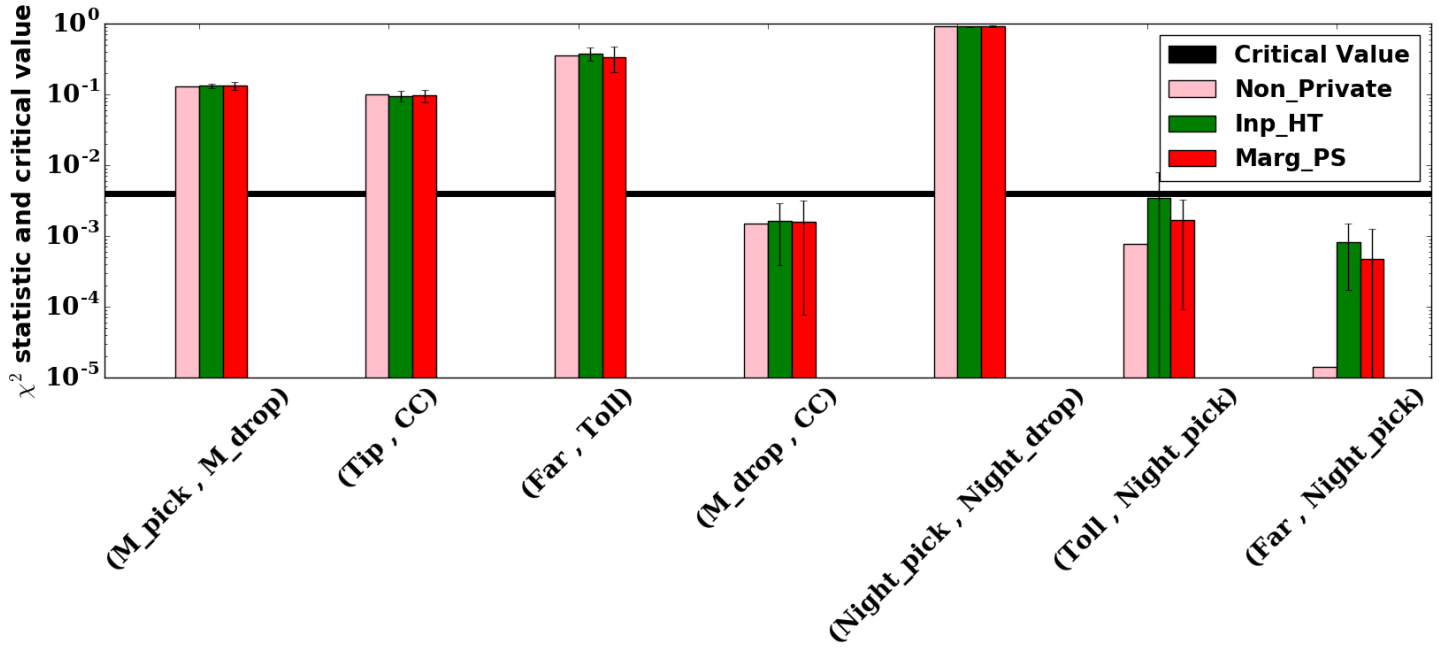
- Create a random variable X_i encoding the error from each user
  - Show that it is unbiased: $E[X_i]=0$, error is zero in expectation
- Compute a bound for its variance, $E[X_i^2]$ (including sampling)
- Use appropriate inequality to bound error of sum, $|\sum_{i=1}^{N} X_i|$
  - Bernstein or Hoeffding in equalities: error like $\sqrt{(N/E[X_i^2])}$
  - Typically, error in average of N goes as $1/\sqrt{N}$
- Possibly, second round of bounding error for further aggregation
  - E.g. first bound error to reconstruct full distribution, then error when aggregating to get a target marginal distribution

Empirical behaviour



- Compare three methods: Hadamard based (Inp_HT), marginal materialization (Marg_PS), Expectation maximization (Inp_EM)
- Measure sum of absolute error in materializing 2-way marginals
- N = 0.5M individuals, vary privacy parameter ε from 0.4 to 1.4

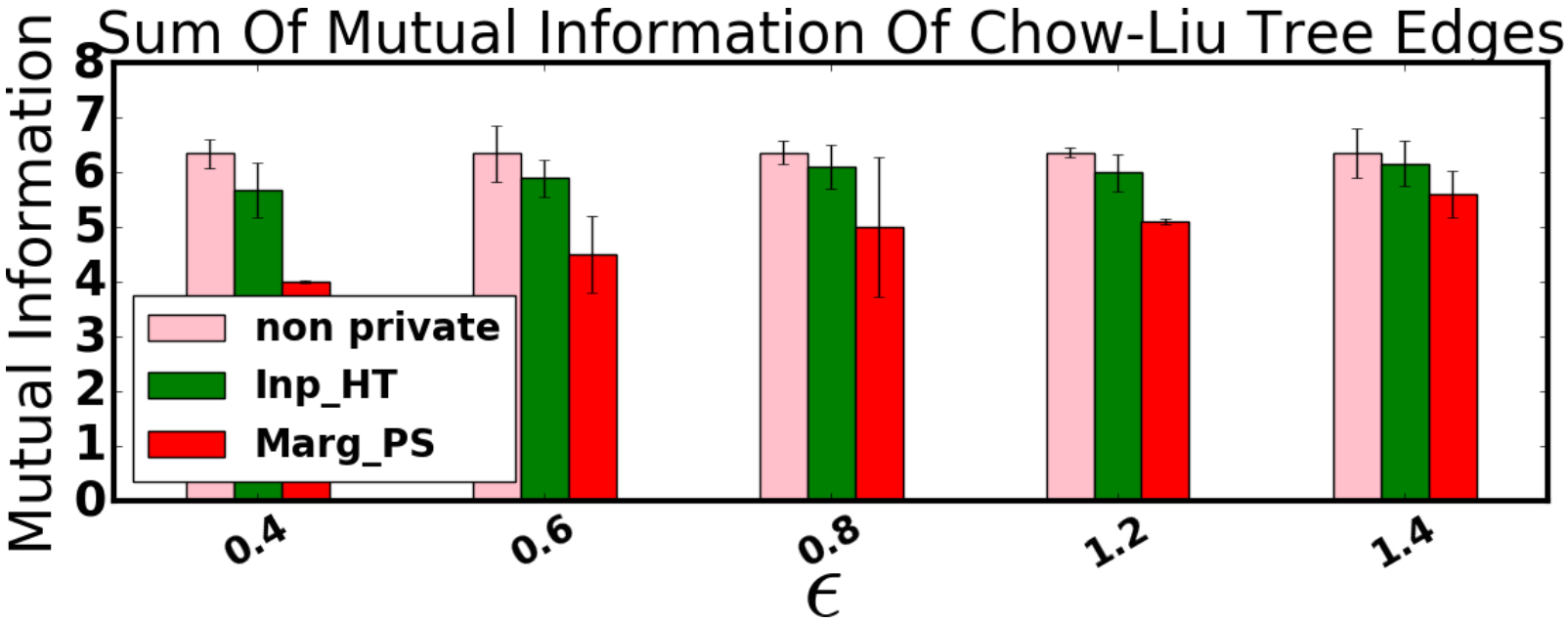
Applications – χ-squared test



- Anonymized, binarized NYC taxi data
- Compute χ-squared statistic to test correlation
- Want to be same side of the line as the non-private value!

THE UNIVERSITY OF WARWICK

Application – building a Bayesian model



- Aim: build the tree with highest mutual information (MI)
- Plot shows MI on the ground truth data for evaluation purposes

Conclusions

- Private data release is a confounding problem!
  - We haven't yet got it right consistently enough
  - The idea of "1 click privacy" is still a long way off
- Current privacy work gives some cause for optimism
  - Statistical privacy, safety in numbers, and massive deployments
- Lots of opportunity for new work:
  - Designing optimal mechanisms for local differential privacy
  - Extend beyond simple counts and marginals
  - Structured data: graphs, movement patterns
  - Unstructured data: text, images, video?

Joint work with Divesh Srivastava (AT&T), Tejas Kulkarni (Warwick)
Supported by AT&T, Royal Society, European Commission