Private Data Analysis over Large Populations

Graham Cormode

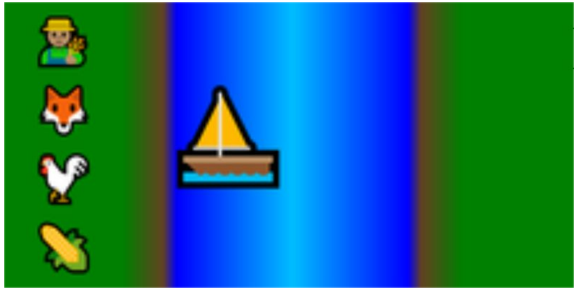
Overview

- The challenges for large scale private data analysis
- Three approaches to private data analysis and recent research results
- Comparison and open questions

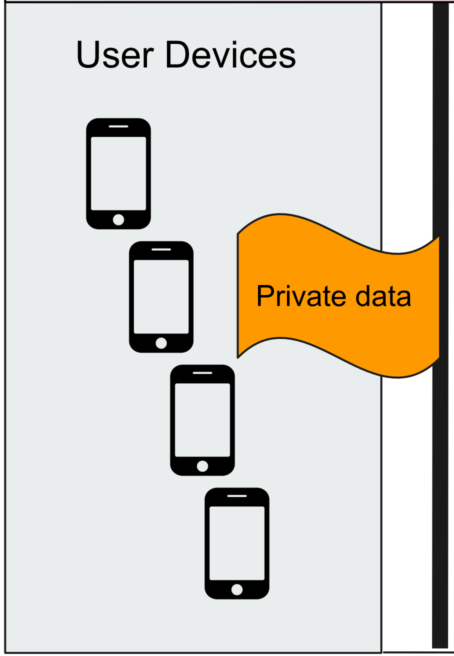
Images are sourced from Wikimedia Commons and attributed accordingly in the notes.

The Private Data Analysis conundrum

- Many large companies have been built on the basis of analyzing data from many users
  - E.g., online advertising, need to understand consumer interests, and provide tailored advertising
- Technology ecosystem changes, new regulations and sensitivity to privacy concerns affect data flow
  - E.g., Apple opt-in data sharing, GDPR/ePD, privacy preservation as a feature
- Conundrum: to understand (sub)population behaviour without compromising individual privacy?
- Canonical example: analyzing user actions in apps on personal devices to central servers



The conundrum: how to bridge this gap?

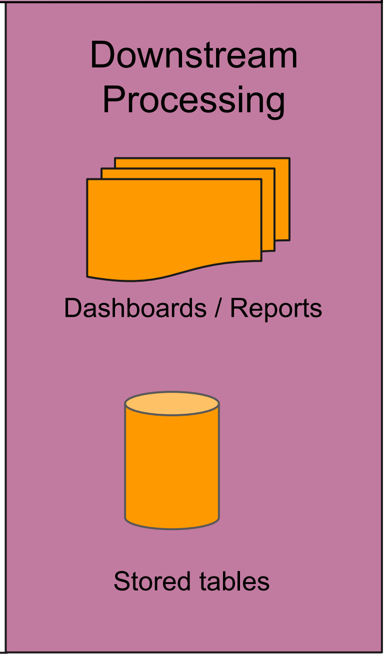


How to get from private data on user devices to private analyses on server-side?

Simply pulling the data across is not a satisfactory option!

Neither is cutting off the flow!

We seek better tradeoffs

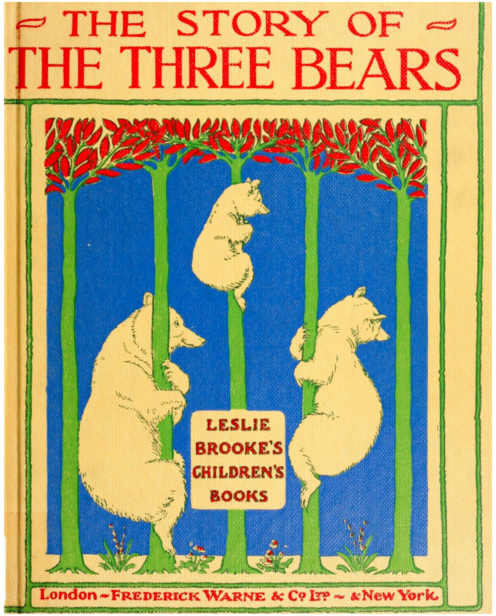


Three possible answers

There is no “right” answer: different solutions achieve tradeoffs between privacy, trust, scalability and cost

This talk outlines three approaches, and mentions research questions relating to each model

- *Federated Analytics (FA)*
- *Privacy preserving query answering (PPQA)*
- *Opt-in debiasing*



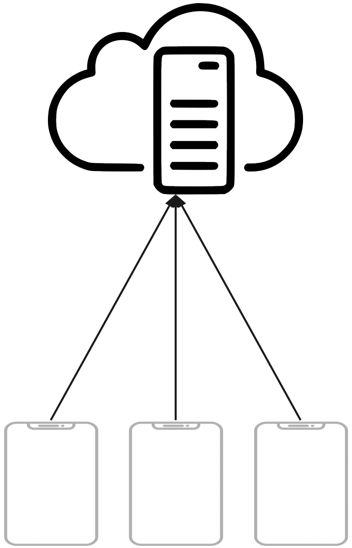
1. The Federated Approach

The federated approach to computation aims to support privacy requirements:

- Data remains under the control of user *clients* (e.g., on their phone)
- Only a small amount of necessary data is shared with central *servers*
- Communication is done under strong security guarantees (e.g., encryption)
- Additional privacy guarantees are provided (e.g., via adding random noise, anonymous communication channels, secure aggregation etc.)

So federated computation is **secure**, **private** and **distributed** (but not fully decentralized)

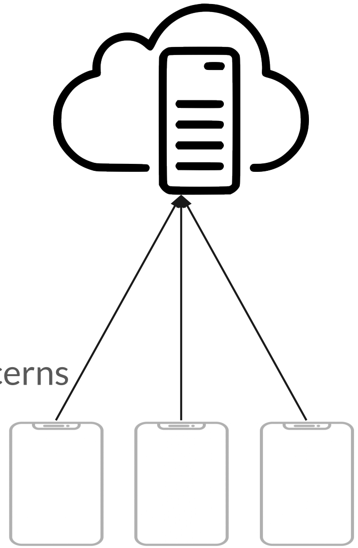
This builds on prior work that achieves subsets of these properties



What is Federated Computation?

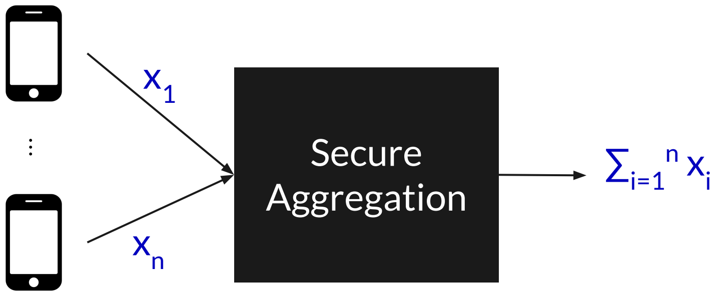
Like MapReduce for highly decentralized data with privacy built in

- Storage is massively distributed (potentially billions of user devices)
- Compute instructions are sent to where the data lives
- Users own and keep their data => Consent, privacy and security are first order concerns
- Non-standard and limited bandwidth/compute/memory availability on nodes
- Intermittent node availability
- Highly ephemeral/unbalanced/non-stationary data



Challenges are to handle the scale of the distributed data, and to provide formal privacy guarantees

Secure Aggregation

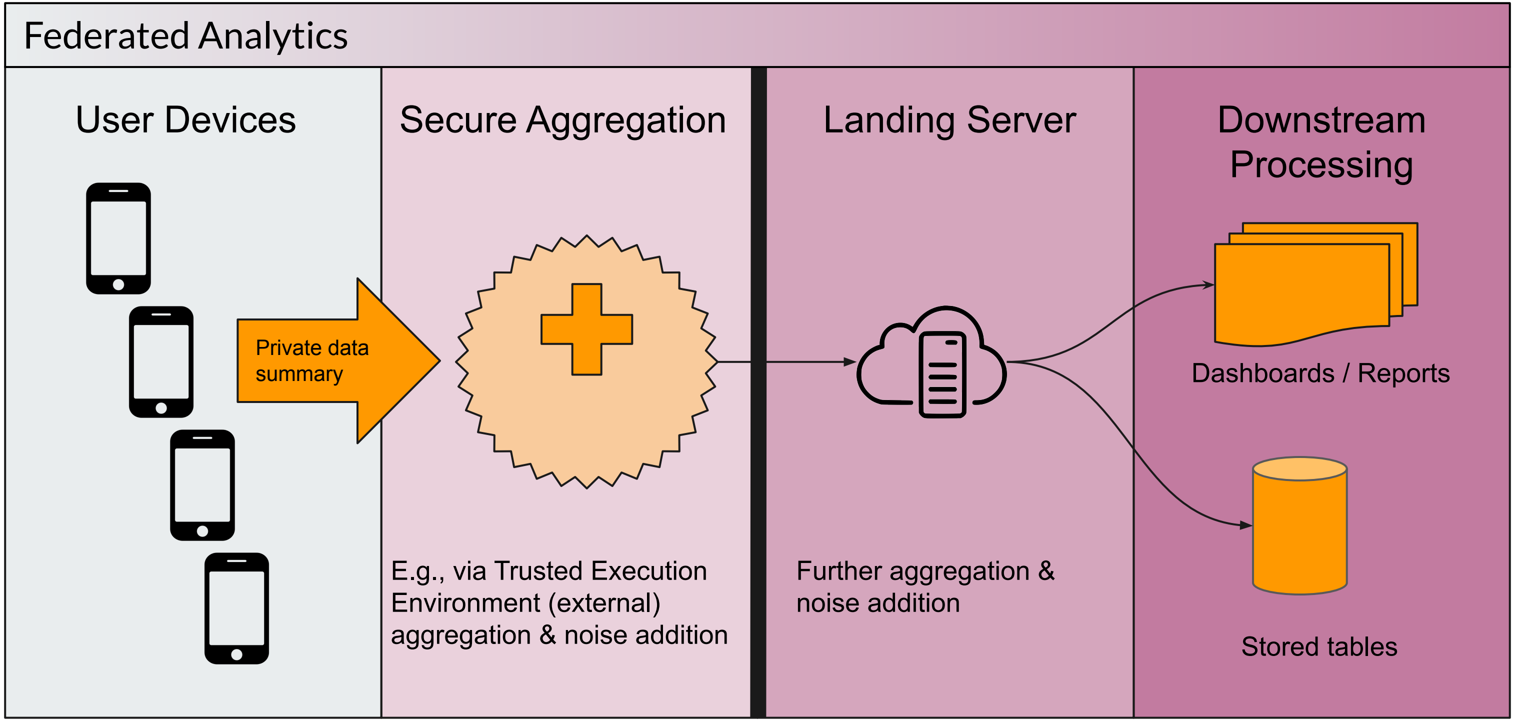


Secure Aggregation addresses the case that we want to compute the sum of vectors held by clients

Various implementations have been proposed with different tradeoffs and trust models:

- Clients secret-share their data to 2 or more servers (SMC-like), who combine the results
- Clients secret-share their data to all other clients, and all pass the shares to a trusted server to aggregate
- Clients secret-share to O(log n) other clients, and all shares are combined by one server
- Clients obtain a "mask" from secure enclave and release data+mask. Enclave sends sum of masks to server
- A subset of clients cooperate to perform cryptographically secure aggregation [Roth et al 19]

Practical implementations emphasize handling client drop-outs: what happens when a client goes offline midway?
Bottom line: we can rely on an implementation of Secure Aggregation to compute sums of input values


Federated Analytics
User Devices
Private data summary
Secure Aggregation
E.g., via Trusted Execution Environment (external) aggregation & noise addition
Landing Server
Further aggregation & noise addition
Downstream Processing
Dashboards / Reports
Stored tables

Federated Analytics

Federated computation focuses on data analytics (as opposed to model training via Federated Learning)

- Core results focus on generating counts, histograms and heavy hitters [AISTATS 22, 23]
- Additional efforts look at various statistics such as mean, variance and median

Recent research results on federated evaluation of classifiers [C., Markov 2023]

- Measuring classifier accuracy shares the same privacy concerns as the core FL training

Federated Post-training statistics

Given a (binary) classifier that has been trained, we want to evaluate:

- **ROC AUC (Area Under Curve)**: a measure of quality of the classifier
- Calibration curves: a function to accurately measure the confidence of a prediction
- Other metrics: the precision, recall, accuracy etc. …

In the federated setting, each client holds examples with a ground truth label (positive or negative)

We show how to capture these via (federated) histogram and quantile primitives

Area Under Curve

Given the score function, we predict x is positive if $s(x) > T$, else negative

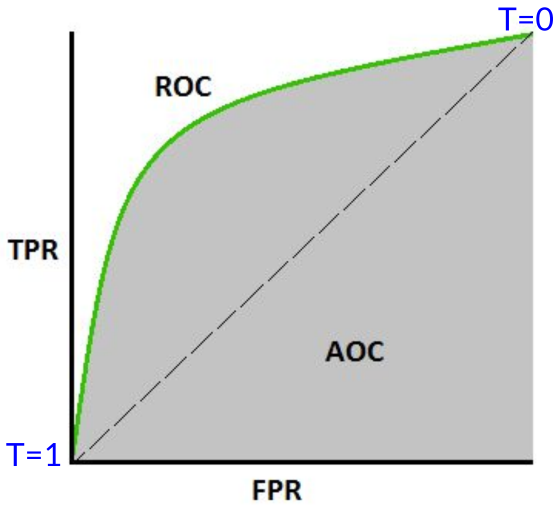
Different choices of T give false positive (FP) / false negative (FN) tradeoffs

Receiver Operator Characteristic curve: plot FPR against TPR as T varies;
Area Under Curve (AUC) measures the tradeoff, between 0.5 and 1.0

Basic calculation: sort examples by score, numerically integrate (quadrature)

But there are equivalent combinatorial calculations:

- Compute sum of ranks of positive examples in sorted scores as S
- $AUC = (S - ½n^+(n^+ - 1)) / (n^+n^-)$, where n^+ (n^-) are the number of positive (negative) examples



Federated Area Under Curve

We make use of histograms to capture information about the classifier behaviour via secure aggregation:

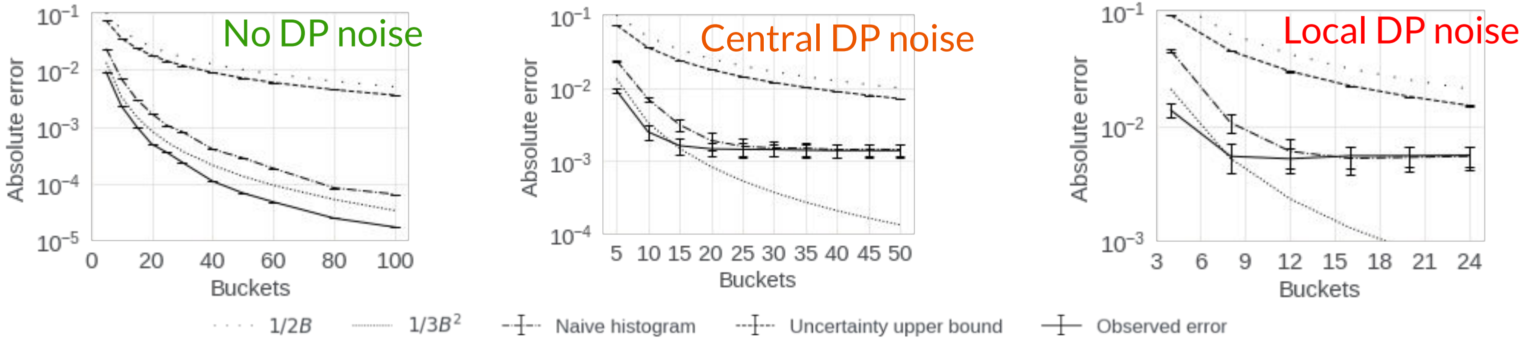
Divide scores into B equal size bins, build **histograms** of number of negatives and positives in each bin

Compute AUC from histogram approximation by one of two (numerically equivalent) options:

a) Treating the bins as piecewise constant score function, and performing quadrature; or
b) Apply the combinatorial calculation based on sum of ranks of positive examples

Error decays as $O(1/B^2)$ under smoothness assumption on score function, or $O((1/B + 1/\varepsilon)1/B)$ with DP

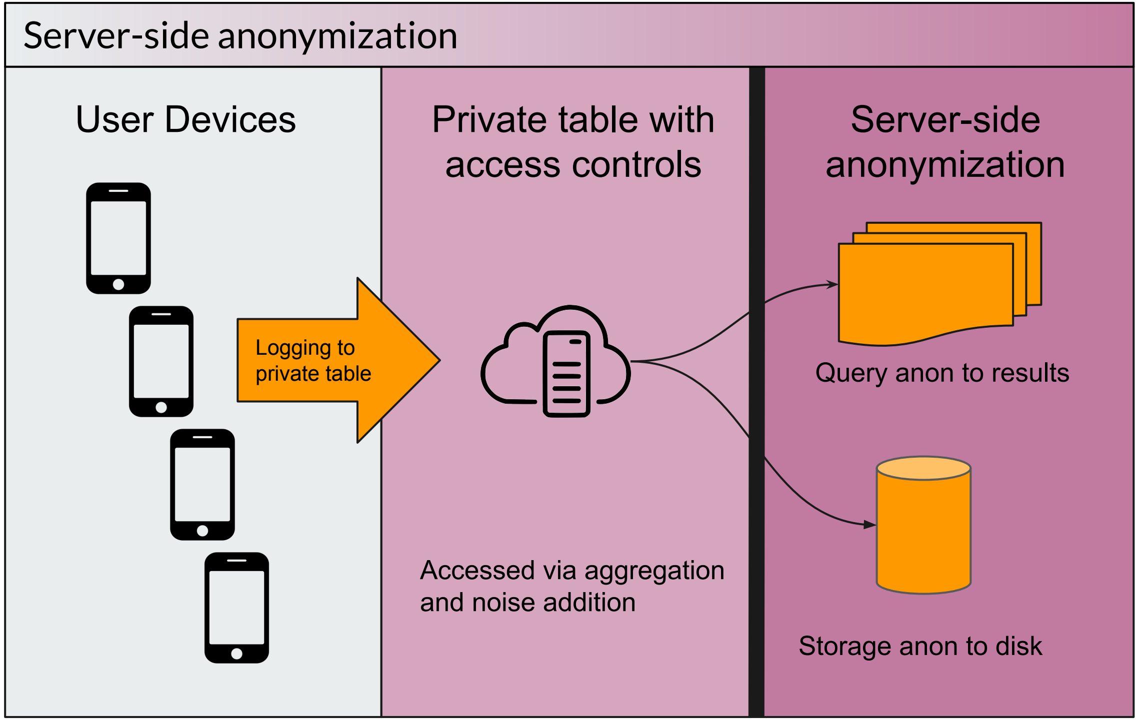
Federated AUC Results



- Error quickly becomes negligible (10^{-3} with 20 buckets, 10^{-4} with 60 buckets) for no noise (left)
- For central DP noise (centre), error plateaus at around 0.002
- 10-20 buckets achieves < 0.005 error for Local DP noise (right)

2. The Server Side Approach

- Gather the data onto a server under strict access controls
  - Permit access to data scientists only via privacy-aware interfaces
  - Allow data scientists to use standard tools e.g., SQL query language
- Ensure that every query result is suitably anonymized
  - E.g., via addition of differentially private noise
- Ensure that queries are isolated to prevent weakening privacy guarantees
- Solution outline: support a limited class of aggregate queries (SUM/COUNT),
  - Automatic query re-writing to add (Laplace/Gaussian) differentially private noise
  - Custom algorithms for specific aggregate functions

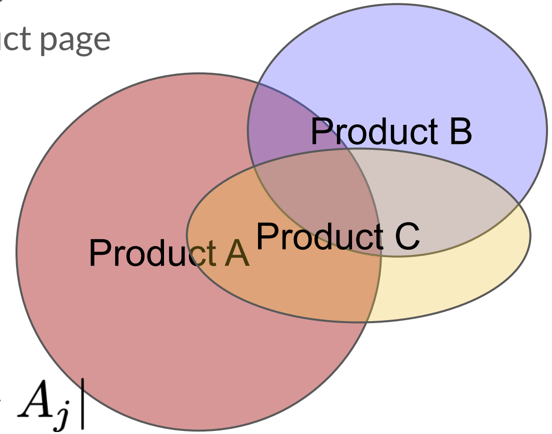

Server-side anonymization
User Devices
Logging to
private table
Private table with
access controls
Accessed via aggregation
and noise addition
Server-side
anonymization
Query anon to results
Storage anon to disk

Server side anonymization

- Many basic operations can be handled easily: SUM, COUNT, SELECT, PROJECT
- But other common tools in the data scientist’s toolbox require extra work:
  - JOIN between tables: need to apply clipping to bound the sensitivity
  - QUANTILES (MIN, MEDIAN, MAX) and other statistical operators require custom solutions
  - COUNT DISTINCT (set cardinality) is a notable example

Approximate distinct counting with merges

- Applications in
  - Business reporting: # unique visitors per demographic group
  - Networking: # unique IP addresses for detecting DDoS attacks
  - Machine learning features: # distinct users that visited a product page

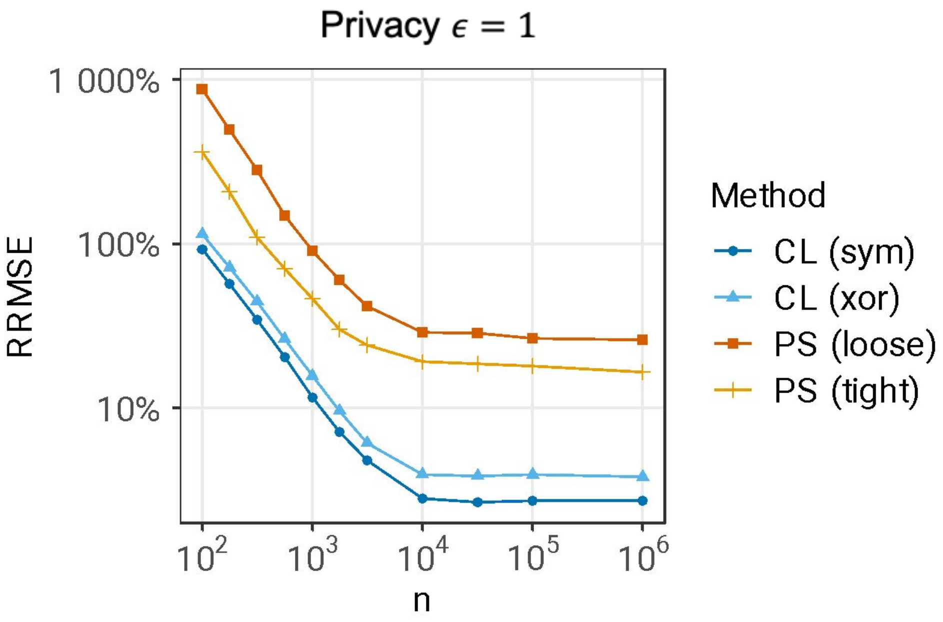


- For each stream of data $A_1, A_2, \ldots, A_k$:
  - Create bounded size summaries S_i that can estimate the number of distinct items in
  - (Basic case): Cardinality of each stream $|A_i|$
  - (Mergeable): Size of any union of a subset of streams $\left| \cup_{j \in \mathcal{J}} A_j \right|$

Private Sketches [Hehir, Ting, C., ICML 2023]

- Existing ‘sketches’ create a summary based on a compact randomized binary encoding
    - E.g., Flajolet-Martin sketches (1983), the Hyperloglog sketch (2007)
- Basic idea: introduce privacy noise by carefully randomly perturbing bits in the sketch
- Can merge private sketches either deterministically or randomized:
    - Deterministic merging: perform ‘exclusive-or’ (XOR) on sketches
    - Randomized merging: optimal merging probability matrix achieves reduced variance
- Likelihood-based estimator provides consistent cardinality estimates
- Implemented in the Presto distributed SQL engine

Empirical results



Sketches

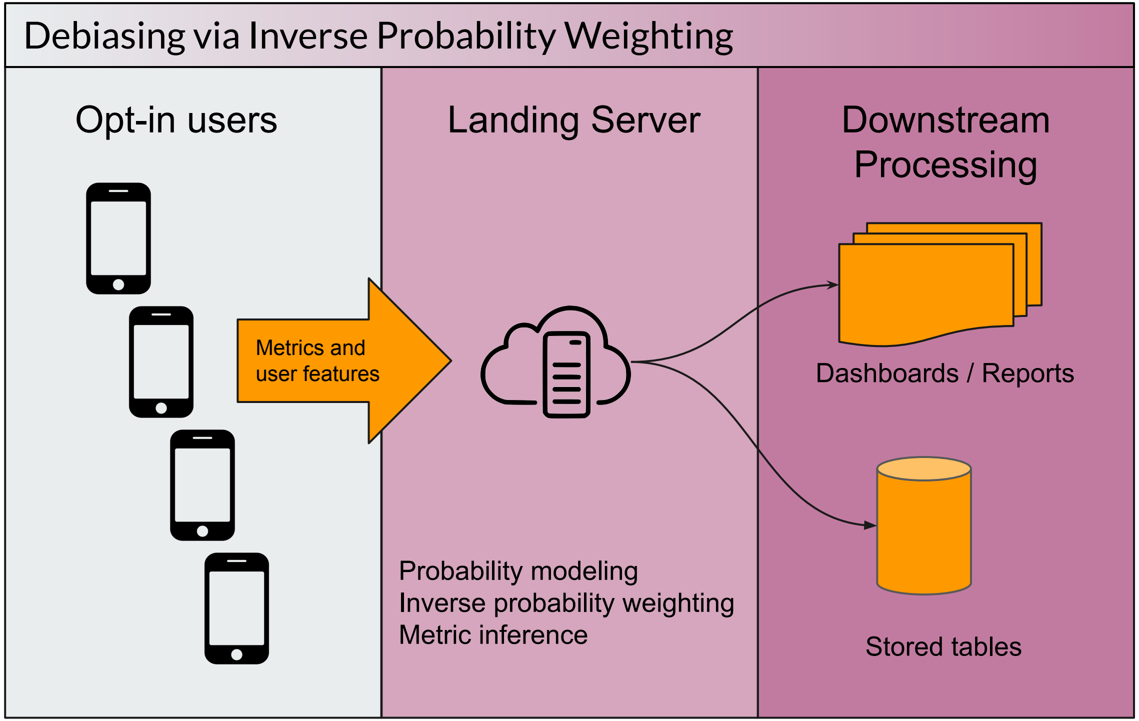
- Baseline: Pagh and Stausholm's sketch with their privacy analysis (loose)
- Our tighter privacy analysis (tight and xor)
- Our Randomized Response sketch (sym)

Two estimators

- Our composite likelihood (CL)
- P&S's estimator (PS)

3. Debiasing Opt-in users

- We can ask users to ‘opt-in’ to private data collection: elect to contribute their data
- Problem: opt-in users are not like other users
  - They tend to be more engaged with the product
  - Demographics do not match the overall population
- Solution: view this as a sampling problem
  - View the opt-in users as a (biased) sample from the overall population
  - Determine appropriate factors to reweight the contributions of the opt-in users
- Approach: build a model to predict likelihood of user opt-in from observable features
  - Determine weights based on the inverse of this propensity score


Debiasing via Inverse Probability Weighting
Opt-in users
Landing Server
Downstream Processing
Metrics and user features
Probability modeling
Inverse probability weighting
Metric inference
Dashboards / Reports
Stored tables

Debiasing challenges

- What model to use to predict opt-in propensity (logistic regression, SVM, NN)?
- How much confidence to place in the debiased statistics? When are they unreliable?
    - E.g., expect poor results on queries correlated with people's privacy preferences
- Does the propensity model need to be built using privacy enhancing technologies?
- How often to rebuild the propensity model?
- How to compare the privacy guarantees to more formal privacy techniques (differential privacy)?

Comparison of approaches

Method	**Pros**	**Cons**
Federated Analytics	Strong privacy guarantees	Higher compute and communication cost
Server-side anonymization	Easier to integrate in existing workflows	Need to trust the server!
Debiasing	No involvement of opt-out users	Currently only empirical accuracy results
…	…	…

Conclusions

No one approach is the perfect solution

Deployed systems may implement multiple of these options

Additional questions arise in practice:

- What extra security tools to use (multiparty computation, secure channels, mix networks)?
- How to debug and monitor secure and private workflows?
- What set of capabilities is sufficient for general purpose analytics?