The Magic of Random Sampling: From Surveys to Big Data

Edith Cohen
Google Research
Tel Aviv University

Disclaimer: Random sampling is classic and well studied tool with enormous impact across disciplines. This presentation is biased and limited by its length, my research interests, experience, understanding, and being a Computer Scientist. I will attempt to present some big ideas and selected applications. I hope to increase your appreciation of this incredible tool.
What is a sample?

A sample is a summary of the data, in the form of a small set of representatives.

Why use samples?

A sample can be an adequate replacement of the full dataset for purposes such as summary statistics, fitting models, other computation/queries.

When to use samples:

- **Inherent Limited availability** of the data
- **Have data but limited resources:**
  - Storage: long or even short term
  - Transmission bandwidth
  - Survey cost of sampled items
  - Computation
  - time delay of processing larger data
History

Basis of (human/animal) learning from observations

Recent centuries: Tool for surveying populations
- **Graunt** 1662: Estimate population of England
- **Laplace** ratio estimator [1786, 1802]: Estimate the population of France. Sampled “Communes” (administrative districts), counting population ratio to live births in previous year. Extrapolate from birth registrations in whole country.
- **Kiaer** 1895 “representative method”; **March** 1903 “probability sampling”
- **US census** 1938: Use probably sample to estimate unemployment

Recent decades: Ubiquitous powerful tool in data modeling, processing, analysis, algorithms design
Sampling schemes/algorithms design goals

A sample is a lossy summary of the data from which we can approximate (estimate) properties of the full data

- Optimize sample-size vs. information content (quality of estimates)
  - Sometimes balance multiple query types/statistics over same sample

- Computational efficiency of algorithms and estimators
  - Efficient on modern platforms (streams, distributed, parallel)
Composable (Mergeable) Summaries

Data $A$

Sample($A$)

Data $B$

Sample($B$)

Data $A \cup B$

Sample($A \cup B$)
Why composable is useful?

- Distributed data/parallelize computation

Streamed data

Sample 1

Sample 2

S. 1 U 2

Sample 5

S. 1 U 2 U 5

Sample 3

Sample 4

S. 3 U 4

1 U 2 U 3 U 4 U 5

Sample(S) → Sample(S) → Sample(S) → Sample(S) → Sample(S) → Sample(S) → Sample(S)
Outline

Selected “big ideas” and example applications

- “Basic” sampling: Uniform/Weighted
- The magic of sample coordination
- Efficient computation over Unaggregated data sets (streamed, distributed)
Uniform (equal probability) sampling

Bernoulli sampling: Each item is selected independently with probability $p$

Reservoir sampling [Knuth 1968, Vitter 1985]: selects a $k$-subset (uniformly at random)
- Fixed sample size $k$; Composable (Knuth’s); state $\propto$ sample size $k$

- Associate with each item $x$ an independent random number $H(x) \sim U[0,1]$
  - Can use a random hash function, limited precision
- Keep $k$ items with smallest $H(x)$

- Correctness: All $k$ subsets have same probability
- Composability: $\text{bottom-}k(A \cup B) = \text{bottom-}k(\text{bottom-}k(A) \cup \text{bottom-}k(B))$
Uniform (equal probability) distinct sampling

Distinct sampling: A (uniform at random) $k$-subset of distinct “keys”

Distinct keys:
Uniform (equal probability) *distinct* sampling

Distinct sampling: A (uniform at random) $k$-subset of *distinct* “keys”
Distinct counting: #$\text{(distinct keys)}$ in (sub)population

**Example Applications:**
- Distinct search queries
- Distinct source-destination IP flows
- Distinct users in activity logs
- Distinct web pages from logs
- Distinct words in a corpus
- ................

**Distinct keys:**

[Images of Pokémon characters]

[Image of a map with circles around Pokémon characters]
Uniform distinct sampling & approximate counting

Distinct sampling: A (uniform at random) $k$-subset of distinct “keys”
Distinct counting: #(distinct keys) in (sub)population

Computation

- (Naïve algorithm) Aggregate then sample/count: state $\propto$ #distinct keys
- Distinct Reservoir sampling/approximate counting [Knuth 1968]+ [Flajolet & Martin 1985]: composable summary with state $\propto$ sample size $k$

- Associate with each item $x$ an independent random hash $H(x) \sim U[0,1]$
- Keep $k$ keys with smallest $H(x)$
Estimation from a uniform sample

Domain/Segment queries: Statistic of a selected segment $H$ of population $X$

*Application*: Logs analysis for marketing, planning, targeted advertising

*Examples*: Approximate the number in Pokémons in our population that are
- College-educated water type
- Flying and hatched in Boston
- Weighing less than 1kg
Estimation from a uniform sample

Domain/Segment queries: Statistic of a selected segment $H$ of population $X$

- **Ratio estimator** (when total population size $|X|$ is known):
  $$\frac{|S \cap H|}{|S|} |X|$$

- **Inverse probability estimator** [Horvitz Thompson 1952] (when $|X|$ is not available, as with distinct reservoir sampling):
  $$\frac{1}{p} |S \cap H| \quad \text{where} \quad p = \text{Prob} \left[ H(x) \leq \min_{y \notin S} H(y) \right] = (k + 1)^{st}-\text{smallest } H(x)$$

**Properties:**
- Unbiased minimum variance estimator
- Coefficient of variation: $\frac{\sigma}{\mu} = \sqrt{\frac{|X|}{|H|k}}$ “relative error”
- Concentration (Bernstein, Chernoff) - probability of large relative error decreases exponentially
Approximate distinct counting of all data elements

Can use distinct uniform sample, but can do better for this special case

HyperLogLog [Flajolet et al 2007]
- Really small structure: optimal size $O(\epsilon^{-2} + \log \log n)$ for CV $\frac{\sigma}{\mu} = \epsilon$; $n$ distinct keys
- **Idea:** for counting, no need to store keys in sample, suffices to use $k = \epsilon^{-2}$ exponents of hashes. Exponents value concentrated so store one and $k$ offsets.

HIP estimators [Cohen ‘14, Ting ‘15]: halve the variance to $\frac{1}{2k}$!
- **Idea:** track an estimate count as the structure (sketch) is constructed. Add inverse probability of modifying structure with each modification
- Applicable with all distinct counting structures
- Surprisingly, better estimator than possible from final structure
Graphs: Estimating cardinalities of neighborhoods, reachability, centralities

Q: number of nodes
- Reachable from
- Within distance 5 from

- Exact computation is $O(|E||V|)$
- Sample-based sketches [C’ 1997]: near-linear $\tilde{O}(|E|)$

Idea:
- Associate independent random $H(v) \sim U[0,1]$ with all nodes
- Compute for each node $u$ a sketch $S(u)$ of the $\epsilon^{-2}$ reachable nodes $v$ with smallest $H(v)$

Applications: Network analytics, social networks, other data with graph representation
Estimating Sparsity structure of matrix products

**Problem:** Compute a matrix product \( B = A_1 A_2 \cdots A_n \quad n \geq 3 \)

- \( A_i \) are sparse (many more zero entries than nonzero entries)

**Q:** find best order (minimize computation) for performing multiplication

- From associativity, \( B = A_1 A_2 A_3 \) can be computed using \((A_1 A_2)A_3\) or \(A_1 (A_2 A_3)\)

- Computation depends on sparsity structure

[C’ 1996]: The sparsity structure of sub-products (number of nonzeros in rows/columns of products) can be approximated in near-linear time (in number of nonzeros in \( \{A_i\} \)).

**Solution:** Preprocess to determine the best order, then perform computation

**Idea:** Define a graph for product by stacking bi-partite graphs \( G_i \) that correspond to nonzeros in \( A_i \). Row/column sparsity of product corresponds to reachability set cardinality.
Unequal probability, Weighted sampling

Keys $x$ can have weights $w_x$

Segment queries: For segment $H \subset X$, estimate

$$w(H) = \sum_{x \in H} w_x \quad \text{(weight $w(H)$ of my Pokémon bag $H$)}$$

Example applications (logs analysis):

- **Key:** video-user view  **weight:** watch time  **Query:** Total watch time segmented by user and video attributes
- **key:** IP network flow  **weight:** bytes transmitted  **Query:** Total bytes transmitted for segment of flows (srcIP, destIP, protocol, port,...)

Applications: billing, traffic engineering, planning
Unequal probability, Weighted sampling

- Key $x$ has weights $w_x$
- Sample $S$ that includes $x$ with probability $p_x$

Segment query: $w(H) = \sum_{x \in H} w_x$

Inverse probability estimator [HT 52]:
$$\hat{w}(H) = \sum_{x \in H \cap S} \frac{w_x}{p_x}$$

- Unbiased (when $p_x > 0$)
- To obtain statistical guarantees on quality we need to sample heavier keys with higher probability.

Poisson Probability Proportional to Size (PPS) [Hansen Hurwitz 1943]

$p_x \propto w_x$ minimizes sum of per-key variance

- $\text{CV} \ \frac{\sigma}{\mu} = \sqrt{\frac{w(X)}{w(H)k}}$ “relative error”, concentration

Robust: If probabilities are approximate, guarantees degrade gracefully
Unequal probability, Weighted sampling

Composable weighted sampling scheme with fixed sample size $k$: Bottom-$k$/Order samples/“weighted” reservoir

- Associate with each key $x$ the value $r(x) = \frac{u_x}{w_x}$, for independent random $u_x \sim D$
- Keep $k$ keys with smallest $r(x)$

Example with $u_x \sim U[0,1]$

<table>
<thead>
<tr>
<th>Key $x$</th>
<th>$w_x$</th>
<th>$u_x$</th>
<th>$r(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>12.4kg</td>
<td>0.22</td>
<td>0.017</td>
</tr>
<tr>
<td></td>
<td>4.0kg</td>
<td>0.12</td>
<td>0.03</td>
</tr>
<tr>
<td></td>
<td>460.0kg</td>
<td>0.31</td>
<td>0.00067</td>
</tr>
<tr>
<td></td>
<td>30.0kg</td>
<td>0.81</td>
<td>0.027</td>
</tr>
<tr>
<td></td>
<td>210.0kg</td>
<td>0.06</td>
<td>0.00029</td>
</tr>
<tr>
<td></td>
<td>10.0kg</td>
<td>0.72</td>
<td>0.072</td>
</tr>
<tr>
<td></td>
<td>8.0kg</td>
<td>0.45</td>
<td>0.05625</td>
</tr>
<tr>
<td></td>
<td>210.0kg</td>
<td>0.57</td>
<td>0.00271</td>
</tr>
</tbody>
</table>

$w_x$ = weight of key $x$; $u_x$ = random value for key $x$; $r(x)$ = rescaled value
Unequal probability, Weighted sampling

Composable weighted sampling scheme with fixed sample size \( k \):
Bottom-\( k \)/Order samples/“weighted” reservoir

- Associate with each key \( x \) the value \( r(x) = \frac{u_x}{w_x} \), for independent random \( u_x \sim D \)
- Keep \( k \) keys with smallest \( r(x) \)

Estimation: Inverse probability
\[
\tau = \text{smallest } r_x \text{ of } x \notin S \quad ; \quad p_x = \text{Prob}[r(x) < \tau] = \text{Prob}_{y \sim D}[y < w_x \tau]
\]

- Without-replacement \( D = \text{EXP}[1] \) [Rosen 72, C’ 97 C’ Kaplan ‘07, Efraimidis Spirakis ’05]
- Priority (sequential Poisson) sampling \( D = U[0,1] \) [Ohlsson ‘00, Duffield Lund Thorup ’07]
  - Essentially same quality guarantees as Poisson PPS

!! works for max-distinct over key value pairs \((e.\text{key}, e.\text{w})\) where \( w_x = \max e.\text{w} \quad e|e.\text{key}=x \)
Unequal probability, Weighted sampling

Example Applications:

Sampling nonnegative matrix products [Cohen Lewis 1997]
Nonnegative matrices $A, B$
Efficiently sample entries in the product $AB$ proportionally to their magnitude (without computing the product)
**Idea:** random walks using edge weight probabilities

Cut/spectral graph sparsifiers of $G = (V, E)$
[Benczur Karger 1996] Sample graph edges and re-weight by inverse probability to obtain a $G' = (V, E')$ such that $|E'| = O(|V|)$ and all cut values are approximately preserved
[Spielman Srivastava 2008] Spectral approximation $L_G \approx L_{G'}$ by using $p_e \propto$(approximate) effective resistance of $e \in E$
Sample Coordination

[Brewer, Early, Joyce 1972]  Same set of keys, multiple sets of weights ("instances"). Make samples of different instance as similar as possible.

Survey sampling motivation: Weight evolve but surveys impose burden. Want to minimize the burden and still have a weighted sample of the evolved set.
## How to coordinate samples

**Coordinated bottom-\(k\) samples:** Use same \(u_x = H(x)\) for all instances

- Associate with each key \(x\) the value \(r(x) = \frac{u_x}{w_x}\), for independent random hash \(u_x \sim D\)
- Keep \(k\) keys with smallest \(r(x)\)

<table>
<thead>
<tr>
<th>Key (x)</th>
<th>Mon (w_x)</th>
<th></th>
<th>Tue (w_x)</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>12.4kg</td>
<td>4.0kg</td>
<td>460.0kg</td>
<td>30.0kg</td>
<td>210.0kg</td>
<td>10.0kg</td>
<td>8.0kg</td>
<td>210.0kg</td>
</tr>
<tr>
<td>Mon (w_x)</td>
<td>1.01v</td>
<td>0.12</td>
<td>0.31</td>
<td>0.81</td>
<td>0.06</td>
<td>0.72</td>
<td>0.45</td>
<td>0.57</td>
</tr>
<tr>
<td>Mon (r(x))</td>
<td>0.017</td>
<td>0.03</td>
<td><strong>0.00067</strong></td>
<td>0.027</td>
<td><strong>0.00029</strong></td>
<td>0.072</td>
<td>0.05625</td>
<td><strong>0.00271</strong></td>
</tr>
<tr>
<td>Tue (r(x))</td>
<td>0.015</td>
<td>0.02</td>
<td><strong>0.00103</strong></td>
<td>0.0162</td>
<td><strong>0.00055</strong></td>
<td>0.144</td>
<td>0.1125</td>
<td><strong>0.0019</strong></td>
</tr>
</tbody>
</table>

!! Change is minimized given that we have a weighted sample of each instance
Coordination of samples

Very powerful tool for big data analysis with applications well beyond what [Brewer, Early, Joyce 1972] could envision

- **Locality Sensitive Hashing (LSH)** (similar weight vectors have similar samples/sketches)
- **Multi-objective samples** (universal samples): A single sample (as small as possible) that provides statistical guarantees for multiple sets of weights/functions.
- **Statistics/Domain queries that span multiple “instances”** (Jaccard similarity, $L_p$ distances, distinct counts, union size, sketches of coverage functions...)
  - MinHash sketches are a special case with 0/1 weights.
- **Facilitates faster computation of samples. Example:** [C’97] Sketching/sampling reachability sets and neighborhoods of all nodes in a graph in near-linear time.
- **Facilitates efficient optimization over samples:** Optimize objective over sets of weights/functions/parametrized functions. Example: Centrality/clustering objective [CCK ‘15], learning [Kingman Welling ‘14]
Multi-objective Sample

- Same keys can have different “weights:” IP flows have bytes, packets, count
- We want to answer segment queries with respect to all weights.
- Naïve solution: 3 disjoint samples
- Smart solution: A single multi-objective sample
Multi-objective sample of $f$-statistics

One set of weights $w_x$, but are interested in *different functions* $f(w_x)$

<table>
<thead>
<tr>
<th>$w_x$</th>
<th>135</th>
<th>2</th>
<th>9</th>
<th>18</th>
<th>21</th>
<th>4</th>
<th>11</th>
<th>4</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>x</code></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Count</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td><code>cap_{5}(w_x)</code></td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td><code>thresh_{10}</code></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Multi-objective Sample  

Set of functions $w \in W$ where $w: X \rightarrow R_{+}$

- Compute coordinated samples $S^{(i)}$ for each $w^{(i)} \in W$

Our multi-objective sample $S$ for $W$ is

- The union $S = \bigcup_i S^{(i)}$
- Sampling probabilities $p_x = \max_i p^{(i)}(x)$

**Theorem:** For any domain query: $w^{(i)}(H) = \sum_{x \in H} w^{(i)}_x$

The inverse probability estimator $\hat{w}^{(i)}(H) = \sum_{x \in H \cap S} \frac{w^{(i)}_x}{p_x}$

Is **unbiased** and provides **statistical guarantees** that are at least as strong as an estimator applied to the dedicated sample $S^{(i)}$
# Multi-objective sample of statistics

One set of weights $w_x$, different functions $f(w_x)$

<table>
<thead>
<tr>
<th>$x$</th>
<th>$w_x$</th>
<th>Count</th>
<th>$f_{\text{cap}_5(w_x)}$</th>
<th>$f_{\text{thresh}_{10}}$</th>
<th>$u_x$</th>
<th>$\frac{u_x}{\text{thresh}_{10}(w_x)}$</th>
<th>$\frac{u_x}{\text{cap}_5(w_x)}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>135</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>0.52</td>
<td>0.104</td>
<td>0.0175</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0.24</td>
<td>0.120</td>
<td>0.41</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>1</td>
<td>5</td>
<td>0</td>
<td>0.76</td>
<td>0.152</td>
<td></td>
</tr>
<tr>
<td></td>
<td>18</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>0.90</td>
<td>0.18</td>
<td></td>
</tr>
<tr>
<td></td>
<td>21</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>0.32</td>
<td>0.064</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>0.14</td>
<td>0.080</td>
<td></td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>0.44</td>
<td>0.088</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0.14</td>
<td>0.0175</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0.07</td>
<td>0.41</td>
<td></td>
</tr>
</tbody>
</table>

For $k = 3$, the MO sample for $F = \{\text{count, thresh}_{10}, \text{cap}_5\}$ is:
Multi-objective sample of all monotone statistics

\( M \): All monotone non-decreasing functions \( f \) with \( f(0) = 0 \)

Examples: \( f(w) = w^p; \ f(w) = \min\{10,w\}; \ f(w) = \log(1 + w) \), .....

Data of key value pairs \((x, w_x)\) : For each \( f \), Instance is \( f(w_x) \) for \( x \in X \)

**Theorem:** [C’97, C’K’07] (threshold functions) [C’15] all \( M \)

Multi-objective sample for all monotone statistics \( M \) has

\begin{itemize}
  \item (expected) sample size: \( O(k \ln n) \), where \( n = \# \text{keys with } w_x > 0 \)
  \item Composable structure of size equal to the sample size
\end{itemize}

\( \Rightarrow \) Very efficient to compute on streamed/parallel/distributed platforms

*Next: Applications to graphs and streams*
Multi-objective sample of monotone statistics

**Theorem:** [C’ 97, C’ K ’07] (threshold functions) [C ‘15] all $M$

Multi-objective sample for all monotone statistics $M$ has

- (expected) sample size: $O(k \ln n)$, where $n = \#\text{keys with } w_x > 0$
- Composable structure of size equal to the sample size

**Application:** Data Streams time-decaying aggregations

monotone non-increasing $\alpha(x)$, and segment $H \subset V$

$$A_\alpha = \sum_{u \in H} \alpha(t_u)$$

- $t_u$: Elapsed time from start of stream to $u$
- $t_u$: Elapsed time from $u$ to current time
Multi-objective sample of monotone statistics

**Theorem:** [C’ 97, C’ K ’07] (threshold functions) [C ‘15] all $M$

Multi-objective sample for all monotone statistics $M$ has

- (expected) sample size: $O(k \ln n)$, where $n = \#\text{keys with } w_x > 0$
- Composable structure of size equal to the sample size

**Application:** Centrality of all nodes in a graph $G = (V, E)$ [C’ 97 C’ Kaplan ‘04 C ‘15]

For a node $v$, monotone non-increasing $\alpha(x)$, and segment $H \subseteq V$

centrality of $v$ for segment $H$ is

$$C_\alpha(v, H) = \sum_{u \in H} \alpha(d_{vu})$$

(Harmonic centrality: $\alpha(x) = \frac{1}{x}$)

**Thm:** All-Distances Sketches (ADS) (MO samples) for all nodes can be computed in $O(|E|)$ computation. We can estimate $C_\alpha(v, H)$ for all $\alpha, H$ from ADS($v$)
Multi-objective sample of distances to a set of points in a metric space [Chechik C’ Kaplan ‘15]

- Metric space $M$
- Set of points $P = \{x_1, x_2, \ldots, x_n\} \subset M$
- Each point $v \in M$ defines weights $w_v(x_i) = d_{v,x_i}$
- A multi objective sample of all $w_v$ allows us to estimate for segments $H \subset P$, and any query point $v$ the sum of distances

$$C(v, H) = \sum_{x_i \in H} d_{v,x_i}$$

**Theorem:**
- Multi-objective overhead for distances is $O(1)$!
- Can be computed using a near-linear number of distance queries

$\implies$ Sample size $O(\epsilon^{-2})$ suffices for estimates of $C(v, P)$ for each $v \in M$ with CV $\epsilon$

!!! Can even relax triangle inequality to $d_{ac} \leq \rho(d_{ab} + d_{bc})$ (e.g. squared distances)
Estimators for multi-instance aggregates

Set of functions $w \in W$ where $w : X \rightarrow R_+$
Coordinated samples $S^{(i)}$ for each $w^{(i)} \in W$

**Example multi-instance aggregations:**
- $L_p^p$ distance $\sum_{x \in H} |w^{(1)}(x) - w^{(2)}(x)|^p$
- One-sided $L_p^p \sum_{x \in H} \max\{0, w^{(1)}(x) - w^{(2)}(x)\}^p$

**Graphs:** Influence estimation from node sketches

**Specific estimators for specific aggregations:**
0/1 weights: Size of union [C’ 95] Jaccard similarity [Broder ‘97]
General weights, tighter estimators: max, min, quantiles [C’ Kaplan 2009, 2011]

**Monotone Estimation Problems** [C’ Kaplan 13, C’ 14]:
- Characterization of all functions for which unbiased bounded variance estimators exist
- Efficient (Pareto optimal) estimators (when they exist)

!!! Coordination is essential in getting good estimators. Independent samples will not work.
Distributed/Streamed data elements: Sampling/counting without aggregation

- Data element \( e \) has key and value \((e.key, e.value)\)
- Multiple elements may have the same key

- Max weight: \( w_x = \max_{e \mid e.key = x} e.value \)
- Sum of weights: \( w_x = \sum_{e \mid e.key = x} e.value \)
- Segment \( f \)-statistics: \( \sum_{x \in H} f(w_x) \)
- \( f \)-statistics: \( \sum_{x \in X} f(w_x) \)
- Sum of weights: \( w_x = \sum_{e \mid e.key = x} e.value \)

- Naïve: Aggregate pairs \((x, w_x)\), then sample - requires state linear in #distinct keys
- Challenge: Sample/Count with respect to \( f(w_x) \) using small state (no aggregation).
  - Sampling gold standard: “aggregated” sample size/quality tradeoffs \( CV = \frac{\sigma}{\mu} = \frac{1}{\sqrt{k}} \)
  - Counting gold standard: like HyperLogLog \( O(\epsilon^{-2} + \log \log n) \), \( CV = \epsilon \)
Distributed/Streamed data elements: Sampling/counting without aggregation

- Data element $e$ has key and value $(e.key, e.value)$
- Multiple elements may have the same key
  - Sum of weights: $w_x = \sum_{e | e.key = x} e.value$
  - Segment $f$-statistics: $\sum_{x \in H} f(w_x)$
  - $f$-statistics: $\sum_{x \in X} f(w_x)$

- Distinct $f(x) = 1$ ($x > 0$): count [Flajolet Martin ‘85, Flajolet et al ‘07] sample [Knuth ‘69]
- Sum $f(x) = x$: count [Morris ‘77], sample [Gibbons Matias ‘98, Estan Varghese ‘05, CDKLT ‘07]
- Frequency moments $f(x) = x^p$: count [Alon Matias Szegedy ‘99, Indyk ‘01]
- “universal” sketches [Braverman Ostrovsky ‘10] (count)

But -- Except for sum and distinct, not even close to ”gold standard”
Sampling/counting without aggregation

- Data element $e$ has key and value $(e.key, e.value)$
- Multiple elements may have the same key
  - Sum of weights: $w_x = \sum_{e | e.key = x} e.value$
  - Segment $f$-statistics: $\sum_{x \in H} f(w_x)$
  - $f$-statistics: $\sum_{x \in X} f(w_x)$

[C’15, C’16] Sampling and counting near “gold standard” $(\times \sqrt{e/(e - 1)} \approx 1.26)$

- $f$: Concave with (sub)linear growth
  - Distinct, sum
  - Low freq. moments: $f(x) = x^p$ for $p \in [0,1]$
  - Capping functions: $Cap_T(x) = \min\{T, x\}$
  - Logarithms: $f(x) = \log(1 + x)$

**Sampling Ideas**: Element process that convert sum to max via distributions to approximate sampling probabilities. Invert the sampling transform for unbiased estimation.

**Counting Ideas**: Element processing guided by Laplace transform to convert to max-distinct approximate counting problem.
Conclusion

We got a taste of sampling “big ideas” that have tremendous impact on analyzing massive data sets.

- Uniform sampling
- Weighted sampling
- Coordination of samples
  - Multi-objective sample
  - Estimation of multi-instance functions
- Sampling and computing statistics unaggregated (distributed/streamed) elements

Future: Still fascinated by sampling and their applications
Near future directions:
- Extend “gold standard” sampling/counting over unaggregated data and understand limits of approach
- Coordination for better mini-batch selection for metric embedding via SGD
- Multi-objective samples for clustering objectives, understand optimization over coordinated samples
Thank you!