Stream Sampling for Frequency Cap Statistics

Edith Cohen

1Google, CA USA

2School of Computer Science
Tel Aviv University, Israel

August 11, 2015
Model: Aggregated / Unaggregated Data

Data *elements* \((x, w)\) have a *key* \(x\) and a numeric *value* \(w > 0\)

- Elements are streamed or distributed, no particular order/partition
- “Unaggregated:” Multiple elements can have the same key
- “Aggregated:” Elements have unique keys
Data elements \((x, w)\) have a key \(x\) and a numeric value \(w > 0\)

- Elements are streamed or distributed, no particular order/partition
- “Unaggregated:” Multiple elements can have the same key
- “Aggregated:” Elements have unique keys

The aggregated view: The set of key value pairs \((x, w_x)\) for active keys \(x\). \(w_x\) is the sum of values of elements with key \(x\).

```
2   2   3   3   2   5
```
Model: Aggregated / Unaggregated Data

Data elements \((x, w)\) have a key \(x\) and a numeric value \(w > 0\):

- Elements are streamed or distributed, no particular order/partition
- “Unaggregated:” Multiple elements can have the same key
- “Aggregated:” Elements have unique keys

The aggregated view: The set of key value pairs \((x, w_x)\) for active keys. \(w_x\) is the sum of values of elements with key \(x\).

Queries are often specified over the aggregated view.
Data elements \((x, w)\) have a key \(x\) and a numeric value \(w > 0\)

- Elements are streamed or distributed, no particular order/partition
- “Unaggregated:” Multiple elements can have the same key
- “Aggregated:” Elements have unique keys

The aggregated view: The set of key value pairs \((x, w_x)\) for active keys \(x\). \(w_x\) is the sum of values of elements with key \(x\).
Data *elements* \((x, w)\) have a *key* \(x\) and a numeric *value* \(w > 0\)

- Elements are streamed or distributed, no particular order/partition
- "Unaggregated:" Multiple elements can have the same key
- "Aggregated:" Elements have unique keys

The *aggregated view*: The set of *key value pairs* \((x, w_x)\) for active keys \(x\).

\(w_x\) is the sum of values of elements with key \(x\).
Data elements \((x, w)\) have a key \(x\) and a numeric value \(w > 0\)

- Elements are streamed or distributed, no particular order/partition
- “Unaggregated:” Multiple elements can have the same key
- “Aggregated:” Elements have unique keys

The aggregated view: The set of key value pairs \((x, w_x)\) for active keys \(x\). \(w_x\) is the sum of values of elements with key \(x\).

Queries are often specified over the aggregated view.
Problem

Computing the aggregated view \( \{(x, w_x)\} \) requires state \( \propto \) number of unique active keys, which can be very large.
Computing over unaggregated data

Problem
Computing the aggregated view \( \{(x, w_x)\} \) requires state \( \propto \) number of unique active keys, which can be very large.

What is state?
- When streaming, the state is what we might keep in memory
- In distributed aggregation, it is the summary size that is shared
Computing over unaggregated data

Problem

Computing the aggregated view \( \{(x, w_x)\} \) requires state \( \propto \) number of unique active keys, which can be very large.

What is state?

- When streaming, the state is what we might keep in memory
- In distributed aggregation, it is the summary size that is shared

Efficient computation requires:

- small state (much smaller than the number of unique keys)
- one (or few) passes over the data
Computing over unaggregated data

Problem

Computing the aggregated view \( \{(x, w_x)\} \) requires state \( \propto \) number of unique active keys, which can be very large.

What is state?

- When streaming, the state is what we might keep in memory
- In distributed aggregation, it is the summary size that is shared

Efficient computation requires:

- small state (much smaller than the number of unique keys)
- one (or few) passes over the data

Why few passes? Historically, Sequential-access storage devices (tape then disks), Unix pipes. Streaming (single pass) is necessary for live dashboards and when data is discarded.

Streaming model: [Knu68], [MG82], [FM85],... formalized in [AMS99]
Frequency statistics

\[ Q(f, H) = \sum_{x \in H} f(w_x) \]

- Function \( f(w) \geq 0 \) for \( w \geq 0 \) so that \( f(0) = 0 \), usually monotone non-decreasing
- Selected segment \( H \subset \mathcal{X} \) (domain, subpopulation) from all keys

Example \( f() \):

- Distinct \( f(w) = 1 \) (# active keys in segment)
- Sum \( f(w) = w \) (sum of weights of keys in segment)
- Moments \( f(w)^p \) with \( p \geq 0 \) (distinct \( p = 0 \), sum \( p = 1 \))
- Cap \( f(w) \equiv \text{cap} T = \min \{ T, w \} \) (distinct \( T = 1 \), sum \( T = +\infty \))

Moments \( w^p \) with \( p \in [0, 1] \) and cap statistics \( \text{cap} T \) with \( T \in (0, +\infty) \) parametrize the range between distinct and sum.
Frequency statistics

\[ Q(f, H) = \sum_{x \in H} f(w_x) \]

- Function \( f(w) \geq 0 \) for \( w \geq 0 \) so that \( f(0) = 0 \), usually monotone non-decreasing
- Selected segment \( H \subset X \) (domain, subpopulation) from all keys

Example \( f() \):
- Distinct \( f(w) = 1 \) (\# active keys in segment)
Frequency statistics

\[ Q(f, H) = \sum_{x \in H} f(w_x) \]

- Function \( f(w) \geq 0 \) for \( w \geq 0 \) so that \( f(0) = 0 \), usually monotone non-decreasing
- Selected segment \( H \subset \mathcal{X} \) (domain, subpopulation) from all keys

Example \( f() \):
- Distinct \( f(w) = 1 \) (# active keys in segment)
- Sum \( f(w) = w \) (sum of weights of keys in segment)
Frequency statistics

\[ Q(f, H) = \sum_{x \in H} f(w_x) \]

- Function \( f(w) \geq 0 \) for \( w \geq 0 \) so that \( f(0) = 0 \), usually monotone non-decreasing
- Selected segment \( H \subset \mathcal{X} \) (domain, subpopulation) from all keys

Example \( f() \):
- **Distinct** \( f(w) = 1 \) (# active keys in segment)
- **Sum** \( f(w) = w \) (sum of weights of keys in segment)
- **Moments** \( f(w) = w^p \) \( (p \geq 0) \) (distinct \( p = 0 \), sum \( p = 1 \))
**Frequency statistics**

\[ Q(f, H) = \sum_{x \in H} f(w_x) \]

- Function \( f(w) \geq 0 \) for \( w \geq 0 \) so that \( f(0) = 0 \), usually monotone non-decreasing
- Selected segment \( H \subset \mathcal{X} \) (domain, subpopulation) from all keys

**Example \( f() \):**

- **Distinct** \( f(w) = 1 \) (\# active keys in segment)
- **Sum** \( f(w) = w \) (sum of weights of keys in segment)
- **Moments** \( f(w) = w^p \) \( (p \geq 0) \) (distinct \( p = 0 \), sum \( p = 1 \))
- **Cap** \( f(w) \equiv \text{cap}_T = \min\{T, w\} \) (distinct \( T = 1 \), sum \( T = +\infty \))
Frequency statistics

\[
Q(f, H) = \sum_{x \in H} f(w_x)
\]

- Function \( f(w) \geq 0 \) for \( w \geq 0 \) so that \( f(0) = 0 \), usually monotone non-decreasing
- Selected segment \( H \subset \mathcal{X} \) (domain, subpopulation) from all keys

Example \( f() \):
- Distinct \( f(w) = 1 \) (# active keys in segment)
- Sum \( f(w) = w \) (sum of weights of keys in segment)
- Moments \( f(w) = w^p \) (distinct \( p = 0 \), sum \( p = 1 \))
- Cap \( f(w) \equiv \text{cap}_T = \min\{T, w\} \) (distinct \( T = 1 \), sum \( T = +\infty \))

Moments \( w^p \) with \( p \in [0, 1] \) and cap statistics \( \text{cap}_T \) with \( T \in (0, +\infty) \) parametrize the range between distinct and sum.
Use case: Frequency capping in online advertising

The first few impressions of the same ad per user are more effective than later ones (diminishing return). Advertisers therefore specify

- A segment of users (based on geography, demographics, other)
- cap the number of impressions per user per time period.

Q: targeted segment: galactic-scale travelers cap: 5
Answer (number of qualifying impressions): 15

Q: targeted segment: non-human intelligent life cap: 3
Answer (number of qualifying impressions): 8
Use case: Frequency capping in online advertising

The first few impressions of the same ad per user are more effective than later ones (diminishing return). Advertisers therefore specify:

- A segment of users (based on geography, demographics, other)
- cap the number of impressions per user per time period.

Q: targeted segment: galactic-scale travelers cap: 5
Answer (number of qualifying impressions): 15

Q: targeted segment: non-human intelligent life cap: 3
Answer (number of qualifying impressions): 8
The first few impressions of the same ad per user are more effective than later ones (diminishing return). Advertisers therefore specify

- A **segment** of users (based on geography, demographics, other)
- **cap** the number of impressions per user per time period.

| 1,567,856 | 22 | 89 | 121 | 2 |

**Q:** targeted segment: **galactic-scale travelers**  cap: **5**

---

**Edith Cohen**  
How to Sample Unaggregated Data
Use case: Frequency capping in online advertising

The first few impressions of the same ad per user are more effective than later ones (diminishing return). Advertisers therefore specify

- A segment of users (based on geography, demographics, other)
- cap the number of impressions per user per time period.

Q: targeted segment: galactic-scale travelers cap: 5
Answer (number of qualifying impressions): 15
Use case: Frequency capping in online advertising

The first few impressions of the same ad per user are more effective than later ones (diminishing return). Advertisers therefore specify

- A **segment** of users (based on geography, demographics, other)
- **cap** the number of impressions per user per time period.

```
|        | 1,567,856 | 22  | 89  | 121 | 2   |
```

Q: targeted segment: **galactic-scale travelers**  cap: 5
Answer (number of qualifying impressions): 15

Q: targeted segment: **non-human intelligent life**  cap: 3
Use case: Frequency capping in online advertising

The first few impressions of the same ad per user are more effective than later ones (diminishing return). Advertisers therefore specify

- A segment of users (based on geography, demographics, other)
- cap the number of impressions per user per time period.

| 1,567,856 | 22 | 89 | 121 | 2 |

Q: targeted segment: galactic-scale travelers  cap: 5
Answer (number of qualifying impressions): 15

Q: targeted segment: non-human intelligent life  cap: 3
Answer (number of qualifying impressions): 8
Advertisers specify:

- A **segment** $H$ of users (based on location, demographics, other)
- A **cap** $T$ on the number of impressions per user per time period.
Advertisers specify:

- A **segment** $H$ of users (based on location, demographics, other)
- A **cap** $T$ on the number of impressions per user per time period.

Campaign planning is interactive. Staging tools use past data to predict the number $Q(cap_T, H)$ of qualifying impressions.

- Data is “unaggregated:” Impressions for same user come from diverse sources (devices, apps, times)
Advertisers specify:

- A **segment** $H$ of users (based on location, demographics, other)
- A **cap** $T$ on the number of impressions per user per time period.

Campaign planning is interactive. Staging tools use past data to predict the number $Q(cap_T, H)$ of qualifying impressions.

- Data is “unaggregated:” Impressions for same user come from diverse sources (devices, apps, times)

$\implies$ Need quick estimates $\hat{Q}(cap_T, H)$ from a summary that is computed efficiently over the unaggregated data set.
Challenges

From the unaggregated data (in one or few passes using small state):

- Basic: Estimate $Q(f, H)$ for a given $f, H \subseteq \mathcal{X}$
- Compute a summary/sample from which we can estimate $Q(f, H)$ for various $f, H$
Frequency statistics challenges

Challenges
From the unaggregated data (in one or few passes using small state):

- Basic: Estimate $Q(f, H)$ for a given $f, H \subseteq \mathcal{X}$
- Compute a summary/sample from which we can estimate $Q(f, H)$ for various $f, H$

Plan for this talk:
**Frequency statistics challenges**

<table>
<thead>
<tr>
<th>Challenges</th>
</tr>
</thead>
<tbody>
<tr>
<td>From the unaggregated data (in one or few passes using small state):</td>
</tr>
<tr>
<td>- Basic: Estimate $Q(f, H)$ for a given $f$, $H \subseteq X$</td>
</tr>
<tr>
<td>- Compute a summary/sample from which we can estimate $Q(f, H)$ for various $f$, $H$</td>
</tr>
</tbody>
</table>

**Plan for this talk:**

- *Aggregated data sets:* The “gold standard” Sample size/ estimation quality tradeoffs.
## Frequency statistics challenges

### Challenges

From the unaggregated data (in one or few passes using small state):

- **Basic**: Estimate $Q(f, H)$ for a given $f$, $H \subseteq X$
- **Compute a summary/sample** from which we can estimate $Q(f, H)$ for various $f$, $H$

### Plan for this talk:

- **Aggregated data sets**: The “gold standard” Sample size/estimation quality tradeoffs.
- **Unaggregated data sets**: How to sample effectively *without* aggregation
Aggregated data: Weighted sampling schemes

- Data elements are key value pairs \((x, w_x)\), elements have unique keys
- Compute a sample \(S_f\) of size \(k\) from which we can estimate \(Q(f, H)\).
Aggregated data: Weighted sampling schemes

- Data elements are key value pairs \((x, w_x)\), elements have unique keys
- Compute a sample \(S_f\) of size \(k\) from which we can estimate \(Q(f, H)\).

To get good size/quality tradeoffs, need (roughly) \(\Pr[x \in S] \propto f(w_x)\):

- **Poisson Probability Proportional to Size (PPS)**: Sample keys independently with \(p_x = \min\{1, kf(w_x) / \sum x f(w_x)\}\)
- **VarOpt** [Cha82, CDL+11]: Dependent PPS for sample size exactly \(k\)
- **Bottom-k/weighted reservoir sampling schemes** [Ros97, CK07]
- **Sequential Poisson (priority)** [Ohl98, DTL07]: Seed \(x\) with \(w_x\)
- **PPS without replacement (ppswor)** [Ros72, Coh97, CK07]: Seed \(x\) with \(1/w_x\)
Data elements are key value pairs \((x, w_x)\), elements have unique keys

Compute a sample \(S_f\) of size \(k\) from which we can estimate \(Q(f, H)\).

To get good size/quality tradeoffs, need (roughly) \(\Pr[x \in S] \propto f(w_x)\):

- **Poisson Probability Proportional to Size (PPS)**: Sample keys independently with \(p_x = \min\{1, \frac{kf(w_x)}{\sum_x f(w_x)}\}\)

- **VarOpt \([Cha82, CDL+11]\)**: Dependent PPS for sample size exactly \(k\)
Aggregated data: Weighted sampling schemes

- Data elements are key value pairs \((x, w_x)\), elements have unique keys
- Compute a sample \(S_f\) of size \(k\) from which we can estimate \(Q(f, H)\).

To get good size/quality tradeoffs, need (roughly) \(\Pr[x \in S] \propto f(w_x)\):

- Poisson Probability Proportional to Size (PPS): Sample keys independently with \(p_x = \min\{1, \frac{k f(w_x)}{\sum_x f(w_x)}\}\)
- VarOpt [Cha82, CDL+11]: Dependent PPS for sample size exactly \(k\)

Bottom-\(k\)/order/weighted reservoir sampling schemes [Ros97, CK07]

```plaintext
foreach key x do
  seed(x) ~ Z[f(w_x)]
S ← k keys with smallest seed(x);
τ ← (k + 1)th smallest seed(x)
```

Edith Cohen How to Sample Unaggregated Data
Aggregated data: Weighted sampling schemes

- Data elements are key value pairs $(x, w_x)$, elements have unique keys.
- Compute a sample $S_f$ of size $k$ from which we can estimate $Q(f, H)$.

To get good size/quality tradeoffs, need (roughly) $\Pr[x \in S] \propto f(w_x)$:

- **Poisson Probability Proportional to Size (PPS):** Sample keys independently with $p_x = \min\{1, \frac{kf(w_x)}{\sum_x f(w_x)}\}$
- **VarOpt [Cha82, CDL+11]:** Dependent PPS for sample size exactly $k$

**Bottom-$k$/order/weighted reservoir sampling schemes [Ros97, CK07]**

```
foreach key x do
    seed(x) \sim Z[f(w_x)]
S \leftarrow k \text{ keys with smallest } seed(x); \quad \tau \leftarrow (k + 1)\text{th smallest } seed(x)
```

- **Sequential Poisson (priority) [Ohl98, DTL07]:** $seed(x) \sim U[0, 1/w_x]$
- **PPS without replacement (ppswor) [Ros72, Coh97, CK07]:**
  $seed(x) \sim \text{Exp}[w_x]$
Inverse probability estimator of $Q(g, H)$ from the sample $S$ [HT52]

$p_x = \Pr[x \in S]$: probability that key $x$ is sampled
For each key $x$, estimate $g(w_x)$ by 0 if $x \not\in S$ and by $g(w_x)/p_x$ if $x \in S$. 

$\hat{Q}(g, H) = \sum_{x \in H} \hat{g}(w_x) = \sum_{x \in H \cap S} g(w_x)/p_x$. 

Applies when we can compute $p_x$ for $x \in S$ nonnegative (since $g$ is)
unbiased (if $g(w_x) > 0 \Rightarrow f(w_x) > 0$) 

Bottom-$k$ samples: $p_x$ is not available so instead we use $p_x|\tau \equiv \Pr[\text{seed}(x) < \tau] = \Pr[Z[f(w_x)] < \tau]$
For ppswor $Z[y] \equiv \text{Exp}[y]$:
$p_x|\tau = 1 - e^{-f(w_x)\tau}$
For priority $Z[y] \equiv \text{U}[0, 1/y]$:
$p_x|\tau = \min\{f(w_x)\tau, 1\}$
Inverse probability estimator of $Q(g, H)$ from the sample $S$ [HT52]

$p_x = \Pr[x \in S]$: probability that key $x$ is sampled

For each key $x$, estimate $g(w_x)$ by 0 if $x \notin S$ and by $g(w_x)/p_x$ if $x \in S$.

$$\hat{Q}(g, H) = \sum_{x \in H} \hat{g}(w_x) = \sum_{x \in H \cap S} \frac{g(w_x)}{p_x}.$$
Aggregated data: Estimators for weighted samples

Inverse probability estimator of $Q(g, H)$ from the sample $S$ [HT52]

$p_x = \Pr[x \in S]$: probability that key $x$ is sampled
For each key $x$, estimate $g(w_x)$ by 0 if $x \not\in S$ and by $g(w_x)/p_x$ if $x \in S$.

$$\hat{Q}(g, H) = \sum_{x \in H} \hat{g}(w_x) = \sum_{x \in H \cap S} \frac{g(w_x)}{p_x}.$$  

Applies when we can compute $p_x$ for $x \in S$
Aggregated data: Estimators for weighted samples

Inverse probability estimator of \( Q(g, H) \) from the sample \( S \) [HT52]

\[
 p_x = \Pr[x \in S]: \text{probability that key } x \text{ is sampled} \\
\text{For each key } x, \text{ estimate } g(w_x) \text{ by } 0 \text{ if } x \not\in S \text{ and by } g(w_x)/p_x \text{ if } x \in S.
\]

\[
 \hat{Q}(g, H) = \sum_{x \in H} \hat{g}(w_x) = \sum_{x \in H \cap S} \frac{g(w_x)}{p_x}.
\]

Applies when we can compute \( p_x \) for \( x \in S \)

- nonnegative (since \( g \) is)  
- unbiased (if \( g(w_x) > 0 \implies f(w_x) > 0 \))
Inverse probability estimator of $Q(g, H)$ from the sample $S$ [HT52]

$p_x = \Pr[x \in S]$: probability that key $x$ is sampled
For each key $x$, estimate $g(w_x)$ by 0 if $x \not\in S$ and by $g(w_x)/p_x$ if $x \in S$.

$$\hat{Q}(g, H) = \sum_{x \in H} \hat{g}(w_x) = \sum_{x \in H \cap S} \frac{g(w_x)}{p_x}.$$  

Applies when we can compute $p_x$ for $x \in S$

- nonnegative (since $g$ is)
- unbiased (if $g(w_x) > 0 \implies f(w_x) > 0$)

**Bottom-k samples:** $p_x$ is not available so instead we use

$$p_x|_\tau \equiv \Pr[\text{seed}(x) < \tau] = \Pr[Z[f(w_x)] < \tau]$$
Aggregated data: Estimators for weighted samples

Inverse probability estimator of $Q(g, H)$ from the sample $S$ \cite{HT52}

\[
\hat{Q}(g, H) = \sum_{x \in H} \hat{g}(w_x) = \sum_{x \in H \cap S} \frac{g(w_x)}{p_x}.
\]

For each key $x$, estimate $g(w_x)$ by $0$ if $x \notin S$ and by $g(w_x)/p_x$ if $x \in S$.

Applies when we can compute $p_x$ for $x \in S$

- nonnegative (since $g$ is)
- unbiased (if $g(w_x) > 0 \implies f(w_x) > 0$)

Bottom-$k$ samples: $p_x$ is not available so instead we use

\[
p_{x|\tau} \equiv \Pr[\text{seed}(x) < \tau] = \Pr[Z[f(w_x)] < \tau]
\]

- For ppswor $Z[y] \equiv \text{Exp}[y]$ : $p_{x|\tau} = 1 - e^{-f(w_x)\tau}$
- For priority $Z[y] \equiv U[0, 1/y]$ : $p_{x|\tau} = \min\{f(w_x)\tau, 1\}$
Aggregated data: Estimators for weighted samples

Inverse probability estimator of $Q(g, H)$ from the sample $S$ [HT52]

$p_x = \Pr[x \in S]$: probability that key $x$ is sampled

For each key $x$, estimate $g(w_x)$ by $0$ if $x \not\in S$ and by $g(w_x)/p_x$ if $x \in S$.

$$\hat{Q}(g, H) = \sum_{x \in H} \hat{g}(w_x) = \sum_{x \in H \cap S} \frac{g(w_x)}{p_x}.$$  

Applies when we can compute $p_x$ for $x \in S$

- nonnegative (since $g$ is)
- unbiased (if $g(w_x) > 0 \implies f(w_x) > 0$)

**Bottom-k samples**: $p_x$ is not available so instead we use

$$p_{x|\tau} \equiv \Pr[\text{seed}(x) < \tau] = \Pr[Z[f(w_x)] < \tau]$$

- For ppswor $Z[y] \equiv \text{Exp}[y] : p_{x|\tau} = 1 - e^{-f(w_x)\tau}$
- For priority $Z[y] \equiv U[0, 1/y] : p_{x|\tau} = \min\{f(w_x)\tau, 1\}$

How good is this estimate?
Let \( q \equiv q(f, H) \) be the fraction of the statistics \( f \) due to segment \( H \):

\[
q = \frac{Q(f, H)}{Q(f, \mathcal{X})} = \frac{\sum_{x \in H} f(w_x)}{\sum_x f(w_x)}.
\]
Let $q \equiv q(f, H)$ be the fraction of the statistics $f$ due to segment $H$:

$$q = \frac{Q(f, H)}{Q(f, \mathcal{X})} = \frac{\sum_{x \in H} f(w_x)}{\sum_x f(w_x)}.$$

bound on the Coefficient of Variation (CV) (relative standard deviation)

$$\sqrt{\frac{\text{var}[\hat{Q}(f, H)]}{Q(f, H)}} \leq \frac{1}{\sqrt{q(k - 1)}}$$
Let $q \equiv q(f, H)$ be the fraction of the statistics $f$ due to segment $H$:

$$q = \frac{Q(f, H)}{Q(f, \mathcal{X})} = \frac{\sum_{x \in H} f(w_x)}{\sum_x f(w_x)}.$$

**bound on the Coefficient of Variation (CV) (relative standard deviation)**

$$\sqrt{\frac{\text{var}[\hat{Q}(f, H)]}{Q(f, H)}} \leq \frac{1}{\sqrt{q(k-1)}}$$

**+concentration:** sample size $k = c \epsilon^{-2}/q$ then prob. of rel. error $> \epsilon$ decreases exponentially in $c$. 
What can we say about estimate quality when $g() \neq f()$?
Aggregated: ppswor estimate quality when $g() \neq f()$

What can we say about estimate quality when $g() \neq f()$?

**Disparity between $g, f$:**

$$\rho(g, f) = \max_{w>0} \frac{g(w)}{f(w)} \max_{w>0} \frac{f(w)}{g(w)}.$$
What can we say about estimate quality when \( g() \neq f() \)?

**Disparity between \( g, f \):**

\[
\rho(g, f) = \max_{w > 0} \frac{g(w)}{f(w)} \max_{w > 0} \frac{f(w)}{g(w)}.
\]

- Disparity is always \( \rho(g, f) \geq 1 \).
- We have \( \rho(g, f) = 1 \iff g = cf \) for some \( c \).
Aggregated: ppswor estimate quality when $g() \neq f()$

What can we say about estimate quality when $g() \neq f()$?

Disparity between $g, f$:

$$
\rho(g, f) = \max_{w > 0} \frac{g(w)}{f(w)} \max_{w > 0} \frac{f(w)}{g(w)}.
$$

- Disparity is always $\rho(g, f) \geq 1$.
- We have $\rho(g, f) = 1 \iff g = cf$ for some $c$.

Lemma

CV of $\hat{Q}(g, H)$ is at most $\left(\frac{\rho}{q(k-1)}\right)^{0.5}$. 

Edith Cohen

How to Sample Unaggregated Data
Summary: Aggregated data “gold standard” sampling

∀f() ≥ 0, with a weighted sample of size k with respect to f(\(w_x\)):

- ∀ segment \(H\): \(\hat{Q}(f, H)\) has CV ≤ \(\sqrt{\frac{1}{q(f, H)k}}\).
- ∀ \(g() \geq 0, H\): \(\hat{Q}(g, H)\) has CV ≤ \(\sqrt{\frac{\rho(g, f)}{q(g, H)k}}\)
∀ \( f() \geq 0 \), with a weighted sample of size \( k \) with respect to \( f(w_x) \):

- ∀ segment \( H \): \( \hat{Q}(f, H) \) has CV \( \leq \sqrt{\frac{1}{q(f, H) k}} \).
- ∀ \( g() \geq 0 \), \( H \): \( \hat{Q}(g, H) \) has CV \( \leq \sqrt{\frac{\rho(g, f)}{q(g, H) k}} \).

Now back to unaggregated data. Desirables:
∀f() ≥ 0, with a weighted sample of size k with respect to f(w_x):

- ∀ segment H: \( \hat{Q}(f, H) \) has CV \( \leq \frac{1}{q(f, H)k} \).
- ∀ g() ≥ 0, H: \( \hat{Q}(g, H) \) has CV \( \leq \frac{\rho(g, f)}{q(g, H)k} \).

Now back to unaggregated data. Desirables:

- **Computation**: One pass (when streaming) or two passes with state proportional to sample size k (\( \implies \) can’t compute aggregated view).
- **Estimation**: Sample size/estimation quality tradeoff close to gold standard.
Toolbox for frequency functions on unaggregated streams

Deterministic algorithms: Misra Gries [MG82] for space saving
Random linear projections (linear sketches): Project vector of key values to a vector with logarithmic dimension. JL transform [JL84] and stable distributions [Ind01] for frequency moments $p \in [0, 2]$.
Sampling-based: Distinct Reservoir Sampling [Knu68] and MinHash sketches [FM85, Coh97] (distinct statistics), Sample and Hold [GM98, EV02, CDK +14] (sum statistics).

No previous solutions for general cap statistics.
Toolbox for frequency functions on unaggregated streams

- **Deterministic algorithms**: Misra Gries: [MG82] Space saving [MAEA05] for heavy hitters
Toolbox for frequency functions on unaggregated streams

- **Deterministic algorithms**: Misra Gries: [MG82] Space saving [MAEA05] for heavy hitters
- **Random linear projections (linear sketches)**: Project vector of key values to a vector with logarithmic dimension. JL transform [JL84] and stable distributions [Ind01] for frequency moments $p \in [0, 2]$. 

Edith Cohen How to Sample Unaggregated Data
Toolbox for frequency functions on unaggregated streams

- **Deterministic algorithms**: Misra Gries: [MG82] Space saving [MAEA05] for heavy hitters
- **Random linear projections (linear sketches)**: Project vector of key values to a vector with logarithmic dimension. JL transform [JL84] and stable distributions [Ind01] for frequency moments $p \in [0, 2]$.
- **Sampling-based**: Distinct Reservoir Sampling [Knu68] and MinHash sketches [FM85, Coh97] (distinct statistics), Sample and Hold [GM98, EV02, CDK+14] (sum statistics)
Toolbox for frequency functions on unaggregated streams

- **Deterministic algorithms**: Misra Gries: [MG82] Space saving [MAEA05] for heavy hitters

- **Random linear projections (linear sketches)**: Project vector of key values to a vector with logarithmic dimension. JL transform [JL84] and stable distributions [Ind01] for frequency moments $p \in [0, 2]$.

- **Sampling-based**: Distinct Reservoir Sampling [Knu68] and MinHash sketches [FM85, Coh97] (distinct statistics), Sample and Hold [GM98, EV02, CDK+14] (sum statistics)

No previous solutions for general cap statistics.
Sampling framework for unaggregated data

Unifies classic schemes for distinct or sum statistics, generalizes bottom-$k$
1. Scores of elements

Scheme is specified by a random mapping $\text{ElementScore}(h)$ of elements $h = (x, w)$ to a numeric score.
Sampling framework for unaggregated data

Unifies classic schemes for distinct or sum statistics, generalizes bottom-\(k\)

1. Scores of elements

Scheme is specified by a random mapping \(\text{ElementScore}(h)\) of elements \(h = (x, w)\) to a numeric score.

Properties of \(\text{ElementScore}\): Distribution depends only on \(x\) and \(w\). Can be dependent for same key, independent for different keys.
Sampling framework for unaggregated data

Unifies classic schemes for **distinct** or **sum** statistics, generalizes bottom-\(k\) sampling.

1. **Scores of elements**
   
   Scheme is specified by a random mapping \(\text{ElementScore}(h)\) of elements \(h = (x, w)\) to a numeric score.

   **Properties of ElementScore**: Distribution depends only on \(x\) and \(w\). Can be dependent for same key, independent for different keys.

2. **Seeds of keys**
   
   The *seed* of a key \(x\) is the minimum score of all its elements.

   \[
   \text{seed}(x) = \min_{h \text{ with key } x} \text{ElementScore}(h)
   \]
Sampling framework for unaggregated data

Unifies classic schemes for distinct or sum statistics, generalizes bottom-k

1. Scores of elements

Scheme is specified by a random mapping ElementScore(h) of elements h = (x, w) to a numeric score.

Properties of ElementScore: Distribution depends only on x and w. Can be dependent for same key, independent for different keys.

2. Seeds of keys

The seed of a key x is the minimum score of all its elements.

\[ \text{seed}(x) = \min_{h \text{ with key } x} \text{ElementScore}(h) \]

3. Sample (S, τ)

\( S \leftarrow \) the k keys with smallest seed(x) (and their seed values)
\( \tau \leftarrow \) the \((k + 1)\)st smallest seed value.
Sampling unaggregated data: Example

Unaggregated data:

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The aggregated view:

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>7</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>
Sampling unaggregated data: Example

Unaggregated data: \((\text{with ElementScore}(h))\)

<table>
<thead>
<tr>
<th>Element</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.06</td>
</tr>
<tr>
<td>2</td>
<td>0.31</td>
</tr>
<tr>
<td>3</td>
<td>0.78</td>
</tr>
<tr>
<td>3</td>
<td>0.12</td>
</tr>
<tr>
<td>2</td>
<td>0.55</td>
</tr>
<tr>
<td>5</td>
<td>0.29</td>
</tr>
</tbody>
</table>

The aggregated view:

<table>
<thead>
<tr>
<th>Element</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
Sampling unaggregated data: Example

Unaggregated data: \( \text{with ElementScore}(h) \)

```
<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>0.06</td>
<td>0.31</td>
<td>0.78</td>
<td>0.12</td>
<td>0.55</td>
</tr>
</tbody>
</table>
```

The aggregated view: with \( \text{seed}(x) \)

```
<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>7</td>
<td>3</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>0.06</td>
<td>0.29</td>
<td>0.12</td>
<td>0.55</td>
<td></td>
</tr>
</tbody>
</table>
```
Sampling unaggregated data: Example

Unaggregated data: \((\text{with ElementScore}(h))\)

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>0.06</td>
<td>0.31</td>
<td>0.78</td>
<td>0.12</td>
<td>0.29</td>
</tr>
</tbody>
</table>

The aggregated view: \((\text{with seed}(x))\)

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>7</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>0.06</td>
<td>0.29</td>
<td>0.12</td>
<td>0.55</td>
</tr>
</tbody>
</table>

Sample of size \(k = 2\):

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>0.06</td>
<td>0.12</td>
</tr>
</tbody>
</table>

\(\tau = 0.29\)
Unaggregated Sampling for distinct and sum

Element scoring for distinct sampling (realizes Reservoir sampling [Knu68] + Hashing [FM85])

\[ \text{ElementScore}(h) = \text{Hash}(x), \text{for random hash } \text{Hash}(x) \sim U[0, 1] \]
Unaggregated Sampling for distinct and sum

Element scoring for distinct sampling (realizes Reservoir sampling [Knu68] + Hashing [FM85])

\[ \text{ElementScore}(h) = \text{Hash}(x), \text{ for random hash } \text{Hash}(x) \sim U[0, 1] \]

**Correctness:** We have \( \text{seed}(x) \equiv \text{Hash}(x) \implies \) the sample is the \( k \) active keys with smallest hash \( \implies \) uniform sample of \( k \) active keys.

Element scoring for Sum (generalized Sample and Hold [GM98, EV02, CCD11, CDK+14])

\[ \text{ElementScore}(h=(x,w)) \sim \text{Exp}[w] \]
Unaggregated Sampling for distinct and sum

Element scoring for distinct sampling (realizes Reservoir sampling [Knu68] + Hashing [FM85])

\[
\text{ElementScore}(h) = \text{Hash}(x), \text{ for random hash } \text{Hash}(x) \sim U[0, 1]
\]

**Correctness:** We have \( \text{seed}(x) \equiv \text{Hash}(x) \implies \) the sample is the \( k \) active keys with smallest hash \( \implies \) uniform sample of \( k \) active keys.

Element scoring for Sum (generalized Sample and Hold [GM98, EV02, CCD11, CDK+14])

\[
\text{ElementScore}(h=(x,w)) \sim \text{Exp}[w]
\]

**Correctness:** Distribution of the minimum of exponential r.v.’s is exponential with sum of parameters:

\[
\text{seed}(x) \sim \min_{\text{elements } (x,w)} \text{Exp}[w] \equiv \text{Exp}[w_x] \implies \text{ppswor wrt } w_x!
\]
Sampling for cap statistics: $\ell$-capped sampling

**Hurdle 1**

To obtain a sample with gold standard quality for $\text{cap}_\ell$, we need element scoring that would result in inclusion probability $p_x$ roughly proportional to $\text{cap}_\ell(w_x)$.
Sampling for cap statistics: $\ell$-capped sampling

**Hurdle 1**
To obtain a sample with gold standard quality for cap$_\ell$, we need element scoring that would result in inclusion probability $p_x$ roughly proportional to cap$_\ell(w_x)$

**Hurdle 2**
Streaming (one pass): Even if we have the “right” sampling probabilities, we do not have exact weights $w_x$ of sampled keys. We need estimators that work with observed counts $c_x$ instead of with $w_x$. 
Obtaining inclusion probabilities roughly proportional to $\text{cap}_\ell(w_x)$

Each key has a base hash $\text{KeyBase}(x) \sim U[0, 1/\ell]$, obtained using $\text{KeyBase}(x) \leftarrow \text{Hash}(x)/\ell$. An element $h = (x, w)$ is assigned a score by first drawing $v \sim \text{Exp}[w]$ and then returning $v$ if $v > 1/\ell$ and $\text{KeyBase}(x)$ otherwise:

**element scoring for $\ell$-capped samples**

\[
\text{ElementScore}(h) = (v \sim \text{Exp}[w]) \leq 1/\ell \text{ if } v > 1/\ell \text{ and } \text{KeyBase}(x) \text{ otherwise:}
\]

The $\text{Exp}[w]$ draws are independent for different elements and independent of $\text{KeyBase}(x)$. 
$\ell$-capped sampling: Hurdle 1

Obtaining inclusion probabilities roughly proportional to $\text{cap}_\ell(w_x)$
Each key has a base hash $\text{KeyBase}(x) \sim U[0, 1/\ell]$, obtained using $\text{KeyBase}(x) \leftarrow \text{Hash}(x)/\ell$. An element $h = (x, w)$ is assigned a score by first drawing $v \sim \text{Exp}[w]$ and then returning $v$ if $v > 1/\ell$ and $\text{KeyBase}(x)$ otherwise:

**element scoring for $\ell$-capped samples**

$$\text{ElementScore}(h) = (v \sim \text{Exp}[w]) \leq 1/\ell \text{ ? } \text{KeyBase}(x) : v$$

The $\text{Exp}[w]$ draws are independent for different elements and independent of $\text{KeyBase}(x)$.

**seed(x) distribution**

$$\text{seed}(x) \sim (v \sim \text{Exp}[w_x]) \leq 1/\ell \text{ ? } U[0, 1/\ell] : v$$
Obtaining inclusion probabilities roughly proportional to \( \text{cap}_\ell(w_x) \)

Each key has a base hash \( \text{KeyBase}(x) \sim U[0, 1/\ell] \), obtained using

\[ \text{KeyBase}(x) \leftarrow \text{Hash}(x)/\ell. \]

An element \( h = (x, w) \) is assigned a score by first drawing \( v \sim \text{Exp}[w] \) and then returning \( v \) if \( v > 1/\ell \) and \( \text{KeyBase}(x) \) otherwise:

\[
\text{ElementScore}(h) = (v \sim \text{Exp}[w]) \leq 1/\ell \quad \text{?} \quad \text{KeyBase}(x) : v
\]

The \( \text{Exp}[w] \) draws are independent for different elements and independent of \( \text{KeyBase}(x) \).

\[
\text{seed}(x) \sim (v \sim \text{Exp}[w_x]) \leq 1/\ell \quad \text{?} \quad U[0, 1/\ell] : v
\]

- For keys with \( w_x \ll \ell \), this is like ppswor wrt \( w_x \)
Obtaining inclusion probabilities roughly proportional to $\text{cap}_\ell(w_x)$
Each key has a base hash $\text{KeyBase}(x) \sim U[0, 1/\ell]$, obtained using
$\text{KeyBase}(x) \leftarrow \text{Hash}(x)/\ell$. An element $h = (x, w)$ is assigned a score by
first drawing $v \sim \text{Exp}[w]$ and then returning $v$ if $v > 1/\ell$ and
$\text{KeyBase}(x)$ otherwise:

<table>
<thead>
<tr>
<th>element scoring for $\ell$-capped samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{ElementScore}(h) = (v \sim \text{Exp}[w]) \leq 1/\ell ? \text{KeyBase}(x) : v$</td>
</tr>
</tbody>
</table>

The $\text{Exp}[w]$ draws are independent for different elements and
independent of $\text{KeyBase}(x)$.

<table>
<thead>
<tr>
<th>seed($x$) distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{seed}(x) \sim (v \sim \text{Exp}[w_x]) \leq 1/\ell ? U[0, 1/\ell] : v$</td>
</tr>
</tbody>
</table>

- For keys with $w_x \ll \ell$, this is like ppswor wrt $w_x$
- For keys with $w_x \gg \ell$, this is like distinct sampling
2-pass estimation quality

First pass computed the sampled keys. Second pass compute $w_x$ for $x \in S$. We can apply the inverse probability estimator.

**Theorem**

The CV of estimating $Q(cap_T, H)$ from an $\ell$-capped sample of size $k$ with exact weights $w_x$ is at most

$$
\left( \frac{e}{e-1} \frac{\max\{T/\ell, \ell/T\}}{q(k-1)} \right)^{0.5}.
$$

$\rho = \max\{T/\ell, \ell/T\}$ is the disparity between cap $\ell$ and cap $T$. Overhead factor of $(e/e - 1)0.5 \approx 1.26$ over aggregated "gold standard."
2-pass estimation quality

First pass computed the sampled keys. Second pass compute $w_x$ for $x \in S$. We can apply the inverse probability estimator.

**Theorem**

The CV of estimating $Q(\text{cap}_T, H)$ from an $\ell$-capped sample of size $k$ with exact weights $w_x$ is at most

$$\left(\frac{e}{e-1} \frac{\rho}{q(k-1)}\right)^{0.5}. $$

\(\rho = \max\{T/\ell, \ell/T\}\) is the disparity between $\text{cap}_\ell$ and $\text{cap}_T$. 

Overhead factor of \( (e/e-1)^{0.5} \approx 1.26 \) over aggregated "gold standard."
2-pass estimation quality

First pass computed the sampled keys. Second pass compute $w_x$ for $x \in S$. We can apply the inverse probability estimator.

**Theorem**

The CV of estimating $Q(\text{cap}_T, H)$ from an $\ell$-capped sample of size $k$ with exact weights $w_x$ is at most

$$\left( \frac{e}{e-1} \frac{\rho}{q(k-1)} \right)^{0.5}.$$ 

- $\rho = \max\{\frac{T}{\ell}, \frac{\ell}{T}\}$ is the disparity between $\text{cap}_\ell$ and $\text{cap}_T$.
- Overhead factor of $(\frac{e}{e-1})^{0.5} \approx 1.26$ over aggregated “gold standard.”
The streaming algorithm maintains an "observed count" $c_x$ for $x \in S$: 

When we process an element $h = (x, w)$ and $x \in S$, we increase $c_x \leftarrow c_x + w$. When the threshold $\tau$ decreases, counts $c_x$ are decreased to simulate the result of sampling with respect to the new threshold. 

$c_x$ is an r.v. with distribution $\sim D[\tau, \ell, w_x]$. Distribution $D$ defines a transform $Y[\tau, \ell]$ from weights $w_x$ to observed counts $c_x$. Our unbiased estimators are derived by applying $f$ to the inverted transform $Y^{-1}$: 

$$\hat{Q}(f, H) = \sum_{x \in H \cap S} \beta(f, \tau, \ell)(c_x).$$

Where $\beta(f, \tau, \ell)(c) \equiv f(c) / \min\{1, \ell \tau\} + f'(c) / \tau$. Applies when $f$ is continuous and differentiable almost everywhere (this includes all monotone functions).
The streaming algorithm maintains an “observed count” $c_x$ for $x \in S$:
- When we process an element $h = (x, w)$ and $x \in S$, we increase $c_x \leftarrow c_x + w$. 
The streaming algorithm maintains an “observed count” $c_x$ for $x \in S$:

- When we process an element $h = (x, w)$ and $x \in S$, we increase $c_x \leftarrow c_x + w$.
- When the threshold $\tau$ decreases, counts $c_x$ are decreased to simulate the result of sampling with respect to the new threshold.
The streaming algorithm maintains an “observed count” $c_x$ for $x \in S$:

- When we process an element $h = (x, w)$ and $x \in S$, we increase $c_x \leftarrow c_x + w$.
- When the threshold $\tau$ decreases, counts $c_x$ are decreased to simulate the result of sampling with respect to the new threshold.

$\Rightarrow$ $c_x$ is an r.v. with distribution $\sim D[\tau, \ell, w_x]$. 

---

Edith Cohen

How to Sample Unaggregated Data
Streaming estimators: Hurdle 2

The streaming algorithm maintains an “observed count” $c_x$ for $x \in S$:

- When we process an element $h = (x, w)$ and $x \in S$, we increase $c_x \leftarrow c_x + w$.
- When the threshold $\tau$ decreases, counts $c_x$ are decreased to simulate the result of sampling with respect to the new threshold.

$\implies c_x$ is an r.v. with distribution $\sim D[\tau, \ell, w_x]$.

Distribution $D$ defines a transform $Y[\tau, \ell]$ from weights $w_x$ to observed counts $c_x$. Our unbiased estimators are derived by applying $f$ to the inverted transform $Y^{-1}$:

$$\hat{Q}(f, H) = \sum_{x \in H \cap S} \beta^{(f, \tau, \ell)}(c_x).$$
The streaming algorithm maintains an “observed count” \( c_x \) for \( x \in S \):

- When we process an element \( h = (x, w) \) and \( x \in S \), we increase \( c_x \leftarrow c_x + w \).
- When the threshold \( \tau \) decreases, counts \( c_x \) are decreased to simulate the result of sampling with respect to the new threshold.

\( \implies c_x \) is an r.v. with distribution \( \sim D[\tau, \ell, w_x] \).

Distribution \( D \) defines a transform \( Y[\tau, \ell] \) from weights \( w_x \) to observed counts \( c_x \). Our unbiased estimators are derived by applying \( f \) to the inverted transform \( Y^{-1} \):

\[
\hat{Q}(f, H) = \sum_{x \in H \cap S} \beta^{(f, \tau, \ell)}(c_x).
\]

Where

\[
\beta^{(f, \tau, \ell)}(c) \equiv f(c) / \min\{1, \ell \tau\} + f'(c) / \tau
\]
The streaming algorithm maintains an “observed count” $c_x$ for $x \in S$:

- When we process an element $h = (x, w)$ and $x \in S$, we increase $c_x \leftarrow c_x + w$.
- When the threshold $\tau$ decreases, counts $c_x$ are decreased to simulate the result of sampling with respect to the new threshold.

$\implies c_x$ is an r.v. with distribution $\sim D[\tau, \ell, w_x]$.

Distribution $D$ defines a transform $Y[\tau, \ell]$ from weights $w_x$ to observed counts $c_x$. Our unbiased estimators are derived by applying $f$ to the inverted transform $Y^{-1}$:

$$\hat{Q}(f, H) = \sum_{x \in H \cap S} \beta^{(f, \tau, \ell)}(c_x) .$$

Where

$$\beta^{(f, \tau, \ell)}(c) \equiv f(c) / \min\{1, \ell \tau\} + f'(c)/\tau$$

* Applies when $f$ is continuous and differentiable almost everywhere (this includes all monotone functions)
Streaming estimator quality

Theorem

The CV of the streaming estimator $\hat{Q}(\text{cap}_T, H)$ applied to an $\ell$-capped sample is upper bounded by

$$\left(\frac{e}{e-1} \left(1 + \max\{\ell/T, T/\ell\}\right) \right)^{0.5} \frac{q(k-1)}{q(k-1)}.$$

Worst-case overhead over aggregated "gold standard."
Streaming estimator quality

Theorem

The CV of the streaming estimator $\hat{Q}(\text{cap}_T, H)$ applied to an $\ell$-capped sample is upper bounded by

$$\left( \frac{e}{e-1} \left( 1 + \max\{\ell/T, T/\ell\} \right) \right)^{0.5} \left( \frac{q(k-1)}{q(k-1)} \right).$$

Worst-case overhead over aggregated “gold standard.”
(pseudo) Code: Fixed-$k$ 2-pass distributed $\ell$-capped sampling

---

// Pass I: Identify $k$ keys in Sample

// Pass I: Thread adds elements to local summary
Sample ← ∅ // Initialize max heap/dict of key seed pairs

foreach element $h = (x, w)$ do
    if $x$ is in Sample then
        Sample[$x$].seed ← min{Sample[$x$].seed, ElementScore($h$)}
    else
        $s$ ← ElementScore($h$)
        if $s < \max\{\text{Sample}[x].seed\}$ then
            Initialize Sample[$x$]
            Sample[$x$].seed ← $s$;
        if $|\text{Sample}| = k + 1$ then
            $y$ ← arg max{Sample[$x$].seed}
            delete Sample[$y$]

// Pass I: Merge two summaries Sample, Sample2

foreach $x \in \text{Sample2}$ do
    if $x$ is in Sample then
        Sample[$x$].seed ← min{Sample[$x$].seed, Sample2[$x$].seed}
    else
        if Sample2[$x$].seed < max{Sample[$x$].seed} then
            Initialize Sample[$x$]
            Sample[$x$].seed ← Sample2[$x$].seed;
        if $|\text{Sample}| = k + 1$ then
            $y$ ← arg max{Sample[$x$].seed}
            delete Sample[$y$]

// Pass II: Compute $w_x$ for keys in Sample

// Pass II: Process elements in thread
foreach $x \in \text{Sample}$ do // Initialize thread
    Sample[$x$].$w$ ← 0

foreach element $h = (x, w)$ do
    if $x \in \text{Sample}$ then
        Sample[$x$].$w$ ← Sample[$x$].$w$ + Sample2[$x$].$w$

// Pass II: Merge two summaries Sample, Sample2

foreach $x \in \text{Sample}$ do
    Sample[$x$].$w$ ← Sample[$x$].$w$ + Sample2[$x$].$w$
(pseudo) Code: Fixed-k stream $\ell$-capped sampling

```plaintext
foreach stream element $(x, w)$ do // Process element
    if $x$ is in Counters then
        Counters[$x$] ← Counters[$x$] + $w$;
    else
        $\Delta \leftarrow -\frac{\ln(1 - \text{rand}(\))}{\max\{\ell^{-1}, \tau\}}$ // $\sim \text{Exp}[\max\{\ell^{-1}, \tau\}]$
        if $\Delta < w$ and $(\tau \ell > 1$ or $\tau \ell \leq 1$ and $\text{KeyBase}(x) < \tau)$ then // insert $x$
            Counters[$x$] ← $w - \Delta$
        if $|\text{Counters}| = k + 1$ then // Evict a key
            if $\tau \ell > 1$ then
                foreach $x \in \text{Counters}$ do
                    $u_x \leftarrow \text{rand}();$ $r_x \leftarrow \text{rand}();$ $z_x \leftarrow \min\{\tau u_x, \frac{-\ln(1-r_x)}{\text{Counters}[x]}\}$ // $x$’s evict threshold
                    if $z_x \leq \ell^{-1}$ then
                        $\bar{z}_x \leftarrow \text{KeyBase}(x)$
                $y \leftarrow \arg\max_{x \in \text{Counters}} z_x; \text{delete } y \text{ from Counters} // \text{key to evict}$
            $\tau^* \leftarrow z_y // \text{new threshold}$
                foreach $x \in \text{Counters}$ do // Adjust counters according to $\tau^*$
                    if $u_x > \max\{\tau^*, \ell^{-1}\}/\tau$ then
                        Counters[$x$] ← $\frac{-\ln(1-r_x)}{\max\{\ell^{-1}, \tau^*\}}$
                $\tau \leftarrow \tau^*; \text{delete } u, r, z, b // \text{deallocate memory}$
            else // $\tau \ell \leq 1$
                $y \leftarrow \arg\max_{x \in \text{Counters}} \text{KeyBase}(x); \text{Delete } y \text{ from Counters} // \text{evict } y$
                $\tau \leftarrow \text{KeyBase}(y) // \text{new threshold}$
        return($\tau; (x, \text{Counters}[x]) \text{ for } x \text{ in Counters}$)
```

Edith Cohen
How to Sample Unaggregated Data
Simulations

CV bounds of $\sqrt{\frac{e\rho}{e-1}/(qk)}$ (2-pass) and $\sqrt{\frac{e}{e-1}(1 + \rho)/(qk)}$ (1-pass) are worst-case upper bounds.

What is the behavior on realistic instances?

- Quantify gain from second pass
- Understand actual dependence on disparity
- Understand Gain from skew (as in aggregated data)

Experiments on Zipf distributions:

- Zipf parameters $\alpha \in [1, 2]$
- Segment=full population
- Swept query cap $T$ and sampling-scheme cap $\ell$. 

Simulation Results for $\ell$-capped samples

Zipf with parameter $\alpha = 2$, sample size $k = 50$, $m = 10^5$ elements. NRMSE (500 reps) of estimating $Q(\text{cap}_T, \mathcal{X})$ from $\ell$-capped sample.

1-pass: $k = 50$, $\alpha = 2$, $m = 100000$, $rep = 500$, NRMSE

<table>
<thead>
<tr>
<th>$\ell$, $T$</th>
<th>1</th>
<th>5</th>
<th>20</th>
<th>50</th>
<th>100</th>
<th>500</th>
<th>1000</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.126</td>
<td>0.159</td>
<td>0.216</td>
<td>0.274</td>
<td>0.326</td>
<td>0.502</td>
<td>0.597</td>
<td>1.061</td>
</tr>
<tr>
<td>1</td>
<td>0.129</td>
<td>0.141</td>
<td>0.192</td>
<td>0.244</td>
<td>0.293</td>
<td>0.449</td>
<td>0.526</td>
<td>0.908</td>
</tr>
<tr>
<td>5</td>
<td>0.193</td>
<td><strong>0.138</strong></td>
<td>0.146</td>
<td>0.173</td>
<td>0.202</td>
<td>0.300</td>
<td>0.353</td>
<td>0.626</td>
</tr>
<tr>
<td>20</td>
<td>0.277</td>
<td>0.169</td>
<td><strong>0.124</strong></td>
<td>0.118</td>
<td>0.125</td>
<td>0.183</td>
<td>0.216</td>
<td>0.377</td>
</tr>
<tr>
<td>50</td>
<td>0.339</td>
<td>0.206</td>
<td>0.140</td>
<td>0.108</td>
<td>0.094</td>
<td>0.096</td>
<td>0.108</td>
<td>0.182</td>
</tr>
<tr>
<td>100</td>
<td>0.390</td>
<td>0.236</td>
<td>0.146</td>
<td><strong>0.107</strong></td>
<td>0.085</td>
<td>0.046</td>
<td>0.034</td>
<td>0.022</td>
</tr>
<tr>
<td>500</td>
<td>0.397</td>
<td>0.250</td>
<td>0.162</td>
<td>0.114</td>
<td>0.092</td>
<td>0.047</td>
<td>0.034</td>
<td>0.012</td>
</tr>
<tr>
<td>1000</td>
<td>0.396</td>
<td>0.232</td>
<td>0.150</td>
<td>0.108</td>
<td><strong>0.083</strong></td>
<td>0.042</td>
<td>0.031</td>
<td><strong>0.011</strong></td>
</tr>
<tr>
<td>10000</td>
<td>0.404</td>
<td>0.244</td>
<td>0.155</td>
<td>0.114</td>
<td>0.085</td>
<td>0.043</td>
<td>0.032</td>
<td>0.012</td>
</tr>
</tbody>
</table>

2-pass: $k = 50$, $\alpha = 2$, $m = 100000$, $rep = 500$, NRMSE

<table>
<thead>
<tr>
<th>$\ell$, $T$</th>
<th>1</th>
<th>5</th>
<th>20</th>
<th>50</th>
<th>100</th>
<th>500</th>
<th>1000</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td><strong>0.125</strong></td>
<td>0.159</td>
<td>0.216</td>
<td>0.274</td>
<td>0.326</td>
<td>0.502</td>
<td>0.597</td>
<td>1.061</td>
</tr>
<tr>
<td>1</td>
<td>0.127</td>
<td>0.139</td>
<td>0.190</td>
<td>0.244</td>
<td>0.293</td>
<td>0.449</td>
<td>0.526</td>
<td>0.908</td>
</tr>
<tr>
<td>5</td>
<td>0.178</td>
<td><strong>0.137</strong></td>
<td>0.144</td>
<td>0.172</td>
<td>0.202</td>
<td>0.300</td>
<td>0.353</td>
<td>0.626</td>
</tr>
<tr>
<td>20</td>
<td>0.235</td>
<td>0.163</td>
<td><strong>0.123</strong></td>
<td>0.116</td>
<td>0.125</td>
<td>0.183</td>
<td>0.216</td>
<td>0.378</td>
</tr>
<tr>
<td>50</td>
<td>0.282</td>
<td>0.184</td>
<td>0.133</td>
<td>0.106</td>
<td>0.093</td>
<td>0.094</td>
<td>0.106</td>
<td>0.181</td>
</tr>
<tr>
<td>100</td>
<td>0.327</td>
<td>0.204</td>
<td>0.140</td>
<td>0.105</td>
<td>0.083</td>
<td>0.041</td>
<td>0.030</td>
<td>0.020</td>
</tr>
<tr>
<td>500</td>
<td>0.321</td>
<td>0.218</td>
<td>0.152</td>
<td>0.114</td>
<td>0.089</td>
<td>0.042</td>
<td>0.030</td>
<td>0.010</td>
</tr>
<tr>
<td>1000</td>
<td>0.322</td>
<td>0.208</td>
<td>0.143</td>
<td><strong>0.105</strong></td>
<td><strong>0.080</strong></td>
<td>0.039</td>
<td>0.028</td>
<td><strong>0.009</strong></td>
</tr>
<tr>
<td>10000</td>
<td>0.326</td>
<td>0.213</td>
<td>0.147</td>
<td>0.109</td>
<td>0.084</td>
<td>0.040</td>
<td>0.028</td>
<td>0.010</td>
</tr>
</tbody>
</table>

Worst-case: $0.14 \times 1.26 \times \sqrt{\rho} \approx 0.17 \sqrt{\rho}$ (2-pass) $0.17 \times \sqrt{1 + \rho}$ (1-pass)
Observations from Simulations

- Actual NRMSE can be much lower than worst-case bound:
  - The \( \sqrt{e/(e - 1)} \) factor over gold standard” is not seen (“worst case” has many keys with \( w_x \approx \ell \)).
  - Error much lower than \( 1/\sqrt{k} \) for skewed distributions with large \( T \approx \ell \).
Observations from Simulations

- Actual NRMSE can be much lower than worst-case bound:
  - The $\sqrt{e/(e-1)}$ factor over gold standard” is not seen (“worst case” has many keys with $w_x \approx \ell$).
  - Error much lower than $1/\sqrt{k}$ for skewed distributions with large $T \approx \ell$.
- Tighter estimates when $\ell \approx T$
Observations from Simulations

- Actual NRMSE can be much lower than worst-case bound:
  - The $\sqrt{e/(e - 1)}$ factor over gold standard” is not seen (”worst case” has many keys with $w_x \approx \ell$).
  - Error much lower than $1/\sqrt{k}$ for skewed distributions with large $T \approx \ell$.

- Tighter estimates when $\ell \approx T$.

- 2-pass estimation quality is within 10% of 1-pass (⇒ use 2-pass to distribute computation but not to improve estimation).
Conclusion

Summary:

- We presented a framework for sampling unaggregated data which unifies and extends classic solutions for distinct and sum statistics.
Summary:
- We presented a framework for sampling unaggregated data which unifies and extends classic solutions for distinct and sum statistics.
- Sampling schemes specified through their element scoring functions

Future:
- Find element scoring functions for "gold standard" sampling of other monotone frequency functions.
- Understand the limitations of our sampling framework.
Conclusion

Summary:

- We presented a framework for sampling unaggregated data which unifies and extends classic solutions for distinct and sum statistics.
- Sampling schemes specified through their element scoring functions.
- First solution for mid-range cap\(_T\) statistics, nearly matches aggregated gold standard.
  - \(\ell\)-capped sample provides unbiased estimates for all frequency statistics and “gold standard” quality for cap\(_T\) statistics with \(T = \Theta(\ell)\).
  - A multi objective sample provides “gold standard” estimate quality for all cap statistics with logarithmic overhead on sample size.

Future:

Find element scoring functions for “gold standard” sampling of other monotone frequency functions. Understand the limitations of our sampling framework.
Conclusion

Summary:
- We presented a framework for sampling unaggregated data which unifies and extends classic solutions for distinct and sum statistics.
- Sampling schemes specified through their element scoring functions
- First solution for mid-range \( \text{cap}_T \) statistics, nearly matches aggregated gold standard
  - \( \ell \)-capped sample provides unbiased estimates for all frequency statistics and “gold standard” quality for \( \text{cap}_T \) statistics with \( T = \Theta(\ell) \).
  - A multi objective sample provides “gold standard” estimate quality for all cap statistics with logarithmic overhead on sample size.

Future:
- Find element scoring functions for “gold standard” sampling of other monotone frequency functions. Understand the limitations of our sampling framework.
Thank you!
Bibliography I

N. Alon, Y. Matias, and M. Szegedy.
The space complexity of approximating the frequency moments.

E. Cohen, G. Cormode, and N. Duffield.
Structure-aware sampling: Flexible and accurate summarization.

Algorithms and estimators for accurate summarization of unaggregated data streams.

Efficient stream sampling for variance-optimal estimation of subset sums.

M. T. Chao.
A general purpose unequal probability sampling plan.

E. Cohen and H. Kaplan.
Summarizing data using bottom-k sketches.

E. Cohen.
Size-estimation framework with applications to transitive closure and reachability.
E. Cohen.
Stream sampling for frequency cap statistics.

Priority sampling for estimating arbitrary subset sums.

C. Estan and G. Varghese.
New directions in traffic measurement and accounting.

P. Flajolet and G. N. Martin.
Probabilistic counting algorithms for data base applications.

P. Gibbons and Y. Matias.
New sampling-based summary statistics for improving approximate query answers.

D. G. Horvitz and D. J. Thompson.
A generalization of sampling without replacement from a finite universe.
P. Indyk.
Stable distributions, pseudorandom generators, embeddings and data stream computation.

W. Johnson and J. Lindenstrauss.
Extensions of Lipschitz mappings into a Hilbert space.

D. E. Knuth.

A. Metwally, D. Agrawal, and A. El Abbadi.
Efficient computation of frequent and top-k elements in data streams.
In ICDT, 2005.

J. Misra and D. Gries.
Finding repeated elements.

E. Ohlsson.
Sequential poisson sampling.
B. Rosén.
Asymptotic theory for successive sampling with varying probabilities without replacement, I. 

B. Rosén.
Asymptotic theory for order sampling.