# New Tools for Scalable Weighted Sampling: Frequency Capping, Multi-Objective, and more

Edith Cohen

[1]Google, CA USA

[2]School of Computer Science
Tel Aviv University, Israel

October 18, 2015

# Data Model

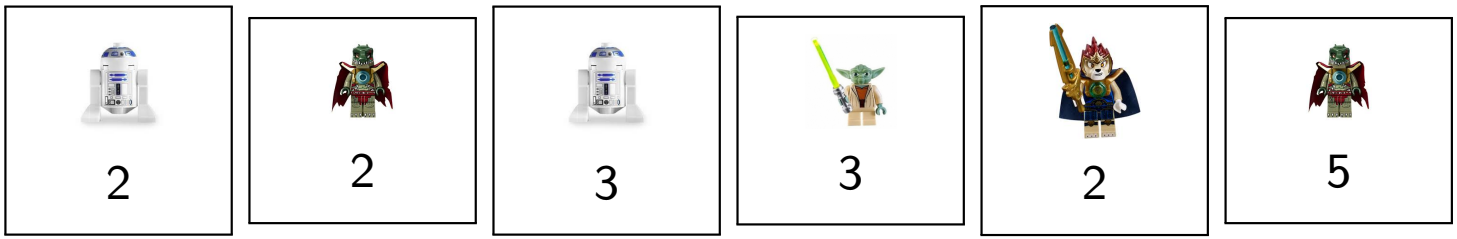
Data *elements* $(x, w)$ have a *key* $x$ and a numeric *value* $w > 0$

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

# Data Model

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

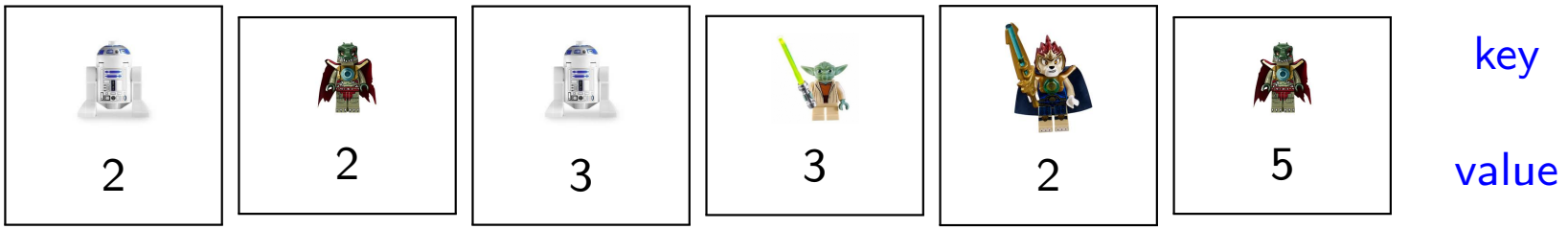
- Elements are streamed or distributed, no particular order/partition
- "Unaggregated:" Multiple elements can have the same key
- "Aggregated:" Elements have unique keys



# Data Model

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

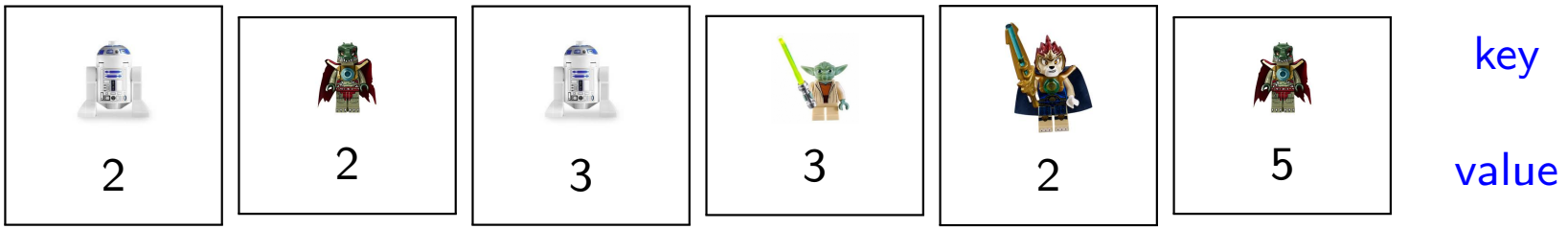
- Elements are streamed or distributed, no particular order/partition
- "Unaggregated:" Multiple elements can have the same key
- "Aggregated:" Elements have unique keys



# Data Model

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

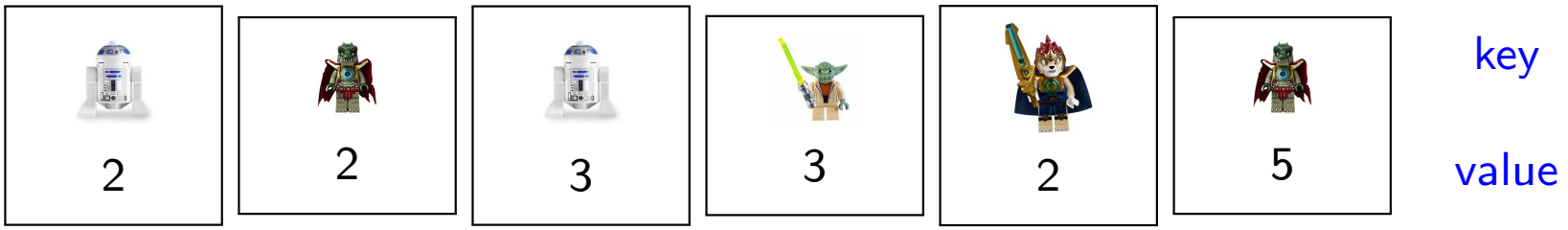



The *aggregated view* of unaggregated data: The set of key value pairs $(x, w_x)$ for active keys $x$. $w_x$ is the sum of values of elements with key $x$.

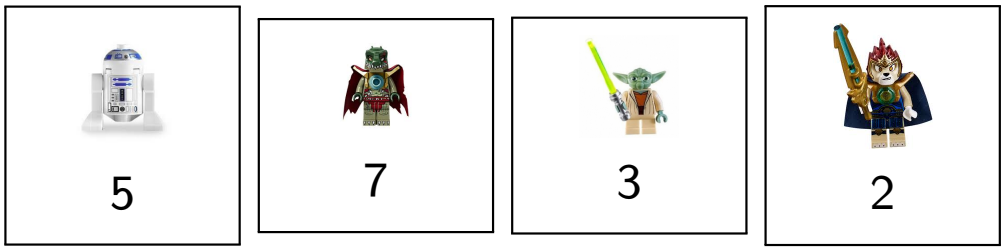
# Data Model

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



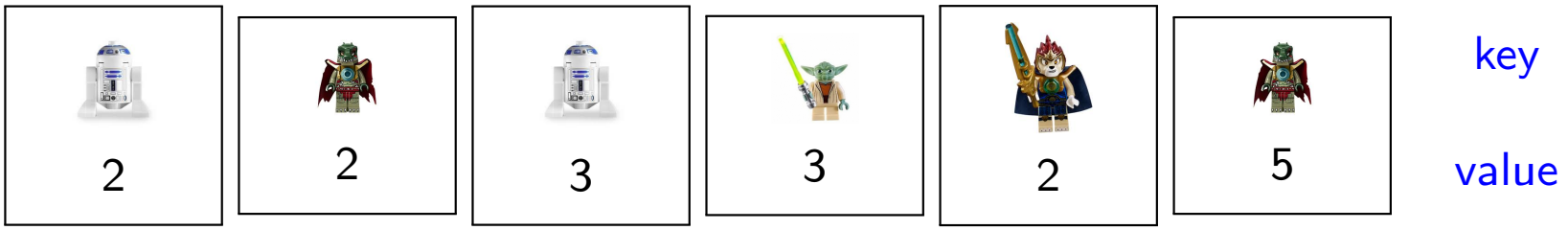

The *aggregated view* of unaggregated data: The set of key value pairs $(x, w_x)$ for active keys $x$. $w_x$ is the sum of values of elements with key $x$.



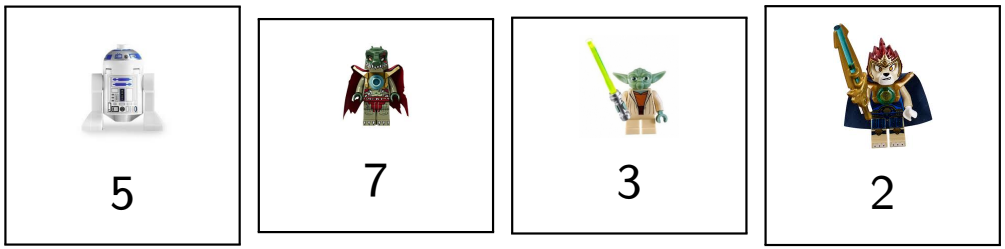
# Data Model

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* of unaggregated data: 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 typically specified over the aggregated view

# Scalable Computation

## One (or few) passes over the data

- Streaming (single sequential pass): Necessary for live dashboards and when data is discarded. Historically model captured sequential-access storage devices (tape, disks), Unix pipes. Streaming model: [Knu68], [MG82], [FM85],. . ., formalized in [AMS99]
- Distributed/Parallel aggregation: Process parts of the data and combine small summaries (look at each part once or few times)

# Scalable Computation

## One (or few) passes over the data

- Streaming (single sequential pass): Necessary for live dashboards and when data is discarded. Historically model captured sequential-access storage devices (tape, disks), Unix pipes. Streaming model: [Knu68], [MG82], [FM85],..., formalized in [AMS99]
- Distributed/Parallel aggregation: Process parts of the data and combine small summaries (look at each part once or few times)

## Small state

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

We want state $\ll$ number of (distinct) keys

# Scalable Computation

## One (or few) passes over the data

- Streaming (single sequential pass): Necessary for live dashboards and when data is discarded. Historically model captured sequential-access storage devices (tape, disks), Unix pipes. Streaming model: [Knu68], [MG82], [FM85],..., formalized in [AMS99]
- Distributed/Parallel aggregation: Process parts of the data and combine small summaries (look at each part once or few times)

## Small state

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

We want state $\ll$ number of (distinct) keys

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

# 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 $f()$:

# 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)

# 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 $p = 1$)
- Capping $f(w) = \mathrm{cap}_T = \min\{T, w\}$ (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 $f(w) = w^p$ $(p \geq 0)$ (distinct $p = 0$, sum $p = 1$)
- Capping $f(w) = \text{cap}_T = \min\{T, w\}$ (distinct $T = 1$, sum $T = +\infty$)
- Threshold $f(w) = \text{thresh}_T = I_{w \geq T}$ $(T > 0)$

# 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 $p = 1$)
- Capping $f(w) = \mathrm{cap}_T = \min\{T, w\}$ (distinct $T = 1$, sum $T = +\infty$)
- Threshold $f(w) = \mathrm{thresh}_T = I_{w \geq T}$ $(T > 0)$

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

# Use case for frequency capping: Online advertising

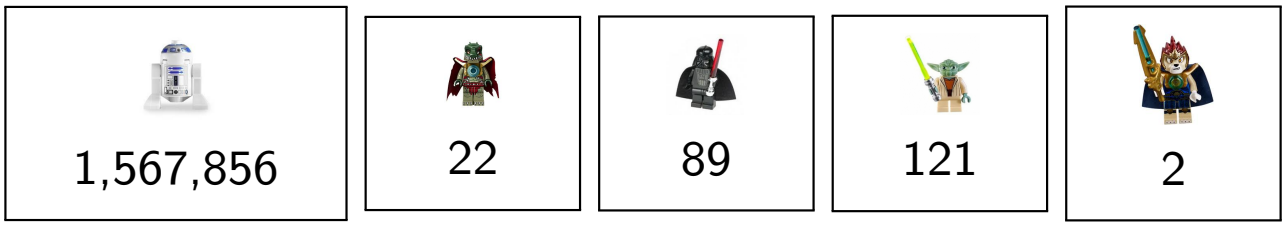
The first few impressions of the same ad per user are more effective than later ones (diminishing return). Advertisers therefore specify

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

# Use case for frequency capping: 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 $T$ on the number of impressions per user per time period.



# Use case for frequency capping: 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 $T$ on the number of impressions per user per time period.

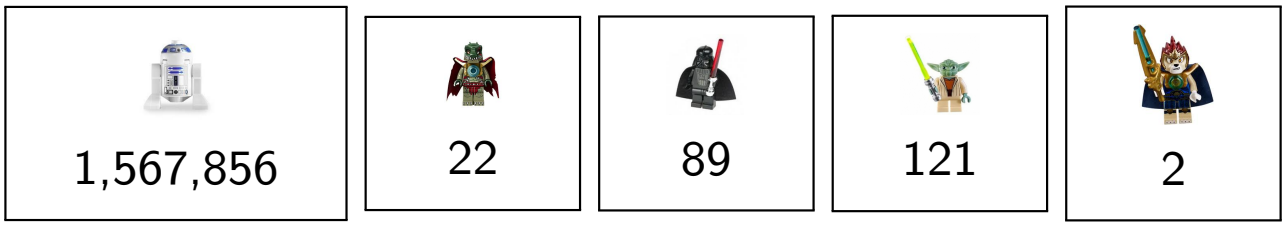
| | | | | |
|---|---|---|---|---|
|  1,567,856 |  22 |  89 |  121 |  2 |

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

# Use case for frequency capping: 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 $T$ on the number of impressions per user per time period.

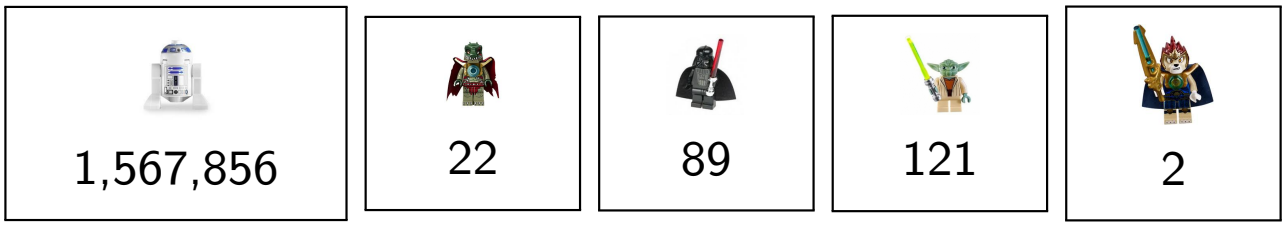



Q: targeted segment: galactic-scale travelers cap: 5
Answer (number of qualifying impressions): 15

# Use case for frequency capping: 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 $T$ on the number of impressions per user per time period.



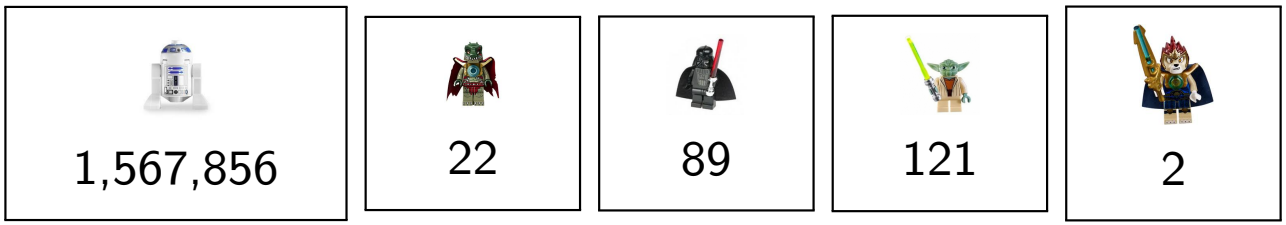

Q: targeted segment: galactic-scale travelers cap: 5
Answer (number of qualifying impressions): 15

Q: targeted segment: non-human intelligent life cap: 3

# Use case for frequency capping: 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 $T$ on 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

# ... Frequency Capping in Online advertising

Advertisers specify:

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

# ... Frequency Capping in Online advertising

Advertisers specify:

- A segment $H$ of users (based on geography, 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 $Q(\text{cap}_T, H)$ of qualifying impressions.

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

# ... Frequency Capping in Online advertising

Advertisers specify:

- A segment $H$ of users (based on geography, 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 $Q(\mathrm{cap}_T, H)$ of qualifying impressions.

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

$\Longrightarrow$ Need quick estimates $\hat{Q}(\mathrm{cap}_T, H)$ from a summary that is computed efficiently over the unaggregated data set.

# Frequency statistics challenges

Multi-objective sample (un)aggregated data: For a set of functions $F$, compute a summary/sample from which we can *estimate* $Q(f, H)$ for various $f \in F$, $H \subseteq \mathcal{X}$.

# Frequency statistics challenges

Multi-objective sample (un)aggregated data: For a set of functions $F$, compute a summary/sample from which we can *estimate* $Q(f, H)$ for various $f \in F$, $H \subseteq \mathcal{X}$.

Weighted sample unaggregated data: For a given $f$, compute a summary/sample from which we can *estimate* $Q(f, H)$ for various $H$

- Basic: Estimate $Q(f, H)$ for a given $f$, $H \subseteq \mathcal{X}$

# Frequency statistics challenges

Multi-objective sample (un)aggregated data: For a set of functions $F$, compute a summary/sample from which we can *estimate* $Q(f, H)$ for various $f \in F$, $H \subseteq \mathcal{X}$.

Weighted sample unaggregated data: For a given $f$, compute a summary/sample from which we can *estimate* $Q(f, H)$ for various $H$

- Basic: Estimate $Q(f, H)$ for a given $f$, $H \subseteq \mathcal{X}$

Goals: • Optimize tradeoffs of sample quality (statistical guarantees) and size. • Scalable computation.

# Frequency statistics challenges

Multi-objective sample (un)aggregated data: For a set of functions $F$, compute a summary/sample from which we can *estimate* $Q(f, H)$ for various $f \in F$, $H \subseteq \mathcal{X}$.

Weighted sample unaggregated data: For a given $f$, compute a summary/sample from which we can *estimate* $Q(f, H)$ for various $H$

- Basic: Estimate $Q(f, H)$ for a given $f$, $H \subseteq \mathcal{X}$

Goals: • Optimize tradeoffs of sample quality (statistical guarantees) and size. • Scalable computation.

Plan for this talk:

# Frequency statistics challenges

Multi-objective sample (un)aggregated data: For a set of functions $F$, compute a summary/sample from which we can *estimate* $Q(f, H)$ for various $f \in F$, $H \subseteq \mathcal{X}$.

Weighted sample unaggregated data: For a given $f$, compute a summary/sample from which we can *estimate* $Q(f, H)$ for various $H$

- Basic: Estimate $Q(f, H)$ for a given $f$, $H \subseteq \mathcal{X}$

Goals: • Optimize tradeoffs of sample quality (statistical guarantees) and size. • Scalable computation.

Plan for this talk:

- *Aggregated data sets*: Review the "gold standard" 

Sample size/ estimation quality tradeoffs. • Multi-objective sampling scheme for all monotone (non-decreasing) $f$. [Coh15b]

# Frequency statistics challenges

Multi-objective sample (un)aggregated data: For a set of functions $F$, compute a summary/sample from which we can *estimate* $Q(f, H)$ for various $f \in F$, $H \subseteq \mathcal{X}$.

Weighted sample unaggregated data: For a given $f$, compute a summary/sample from which we can *estimate* $Q(f, H)$ for various $H$

- Basic: Estimate $Q(f, H)$ for a given $f$, $H \subseteq \mathcal{X}$

Goals: • Optimize tradeoffs of sample quality (statistical guarantees) and size. • Scalable computation.

Plan for this talk:

- *Aggregated data sets*: Review the "gold standard" 

Sample size/ estimation quality tradeoffs. • Multi-objective sampling scheme for all monotone (non-decreasing) $f$. [Coh15b]
- Unaggregated data sets: How to sample effectively *without* aggregation for capping statistics (and more) [Coh15c]

# Aggregated data: Weighted sampling schemes

- Data provided as key value pairs $(x, w_x)$.
- Compute a sample $S_f$ of size $k$ from which we can estimate $Q(f, H)$.

# Aggregated data: Weighted sampling schemes

- Data provided as key value pairs $(x, w_x)$.
- Compute a sample $S_f$ of size $k$ from which we can estimate $Q(f, H)$.

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

# Aggregated data: Weighted sampling schemes

- Data provided as key value pairs $(x, w_x)$.
- 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 provided as key value pairs $(x, w_x)$.
- 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** // $Z[w]$: distribution parameterized by $w$
  $\mathtt{seed}(x) \sim Z[f(w_x)]$

$S \leftarrow k$ keys with smallest $\mathtt{seed}(x)$; $\tau \leftarrow (k+1)$th smallest $\mathtt{seed}(x)$

# Aggregated data: Weighted sampling schemes

- Data provided as key value pairs $(x, w_x)$.
- 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** // $Z[w]$: distribution parameterized by $w$
  $\mathtt{seed}(x) \sim Z[f(w_x)]$
$S \leftarrow k$ keys with smallest $\mathtt{seed}(x)$; $\tau \leftarrow (k+1)$th smallest $\mathtt{seed}(x)$

- Sequential Poisson (priority) [Ohl98, DTL07]:
  $\mathtt{seed}(x) \sim U[0, 1/f(w_x)]$
- PPS without replacement (ppswor) [Ros72, Coh97, CK07]:
  $\mathtt{seed}(x) \sim \mathrm{Exp}[f(w_x)]$

# Aggregated: More on pps without replacement (ppswor)

Two equivalent formulations [Ros72]

**foreach** *key* $x$ **do**
  $\texttt{seed}(x) \sim \text{Exp}[f(w_x)]$
$S \leftarrow k$ keys with smallest $\texttt{seed}(x)$

$S \leftarrow \emptyset$
**repeat**
  Sample $x \notin S$ using

$$p_x = \frac{f(w_x)}{\sum_{y \notin S} f(w_y)}$$

  $S \leftarrow S \cup \{x\}$
**until** $|S| = k$;

# Aggregated: More on pps without replacement (ppswor)

Two equivalent formulations [Ros72]

**foreach** *key* $x$ **do**
  $\texttt{seed}(x) \sim \text{Exp}[f(w_x)]$
$S \leftarrow k$ keys with smallest $\texttt{seed}(x)$

$S \leftarrow \emptyset$
**repeat**
  Sample $x \notin S$ using

$$p_x = \frac{f(w_x)}{\sum_{y \notin S} f(w_y)}$$

  $S \leftarrow S \cup \{x\}$
**until** $|S| = k$;

We focus on ppswor:

- Similar (near optimal) sample size/quality tradeoffs to other weighted sampling schemes
- Our proposed schemes for unaggregated data build on ppswor

# 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 $x \notin S$ and by $g(w_x)/p_x$ if $x \in S$.

# Aggregated data: Estimators for weighted samples

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

$p_x = \Pr[x \in S]$: 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 $x \notin S$ and by $g(w_x)/p_x$ if $x \in S$.

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

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]

$p_x = \Pr[x \in S]$: 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 $x \in S$.

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

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

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

# 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 $x \notin S$ and by $g(w_x)/p_x$ if $x \in S$.

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

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

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

Poisson PPS samples: $p_x = \min\{1, \frac{kf(w_x)}{\sum_x f(w_x)}\}$
We have $w_x$ for sampled keys $x \in S$, and the total $\sum_x f(w_x)$
$\implies$ can compute $p_x$ and apply estimator.

# 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 $x \notin S$ and by $g(w_x)/p_x$ if $x \in S$.

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

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

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

# Aggregated data: Estimators for weighted samples

## Inverse probability estimator of $Q(g, H)$ from the sample $S$ [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 $x \in S$.

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

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

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

The inclusion probability of $x$ *conditioned* on randomization of all other keys:$\tau$ is the $k$th smallest $\mathtt{seed}(y)$ for $y \neq x$; $x \in S \iff \mathtt{seed}(x) < \tau$

- For ppswor $Z[y] \equiv \mathrm{Exp}[y]$ : $p_{x|\tau} = 1 - e^{-f(w_x)\tau}$
- 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 $x \notin S$ and by $g(w_x)/p_x$ if $x \in S$.

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

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

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

# Aggregated data: Estimators for weighted samples

## Inverse probability estimator of $Q(g,H)$ from the sample $S$ [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 $x \in S$.

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

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

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

$$\hat{Q}(g,H) = \sum_{x \in H \cap S} \hat{g}(w_x \mid \tau) \ , \text{ where } \hat{g}(w_x \mid \tau) = \frac{g(w_x)}{p_{x|\tau}} \ .$$

How good is this estimate?

# Aggregated: ppswor estimate quality when $g() = f()$

Let $q \equiv q(f, H)$ be the fraction of the statistics $f$ due to segment $H$:

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

# Aggregated: ppswor estimate quality when $g() = f()$

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)**

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

# Aggregated: ppswor estimate quality when $g() = f()$

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)**

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

+concentration: sample size $k = c\epsilon^{-2}/q$ then prob. of rel. error $> \epsilon$ decreases exponentially in $c$.

# Aggregated: Interpreting the CV bound for $g() = f()$

## CV (relative standard deviation, NRMSE) bound

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

# Aggregated: Interpreting the CV bound for $g() = f()$

## CV (relative standard deviation, NRMSE) bound

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

$\Longrightarrow$ If we want CV $\leq \epsilon$ on segments $H$ that have $q(f,H) \geq q$ fraction of the total $f$ statistics, we need a sample of size $k = \epsilon^{-2}/q$

# Aggregated: Interpreting the CV bound for $g() = f()$

**CV (relative standard deviation, NRMSE) bound**

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

$\Longrightarrow$ If we want CV $\leq \epsilon$ on segments $H$ that have $q(f,H) \geq q$ fraction of the total $f$ statistics, we need a sample of size $k = \epsilon^{-2}/q$

!! This is the optimal size/quality tradeoff for sampling (on average over segments with proportion $q$)

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

For CV $\epsilon \leq 10\%$ and $q \geq 0.1\%$ $\Longrightarrow$ Sample size $k = 10^5$.

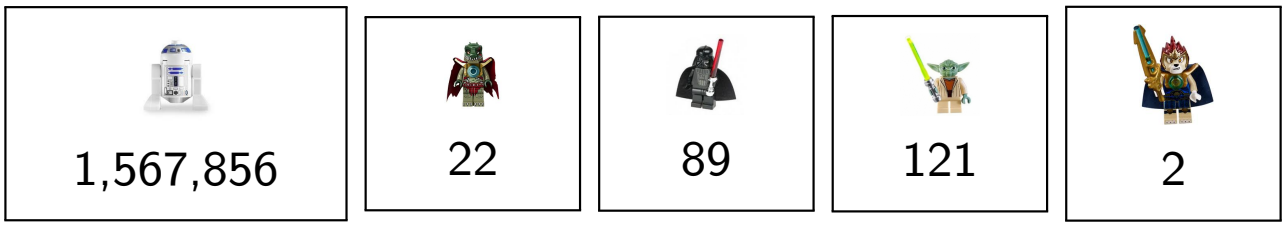
# Aggregated: Interpreting the CV bound for $g() = f()$

**CV (relative standard deviation, NRMSE) bound**

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

$\Longrightarrow$ If we want CV $\leq \epsilon$ on segments $H$ that have $q(f,H) \geq q$ fraction of the total $f$ statistics, we need a sample of size $k = \epsilon^{-2}/q$

!! This is the optimal size/quality tradeoff for sampling (on average over segments with proportion $q$)




For CV $\epsilon \leq 10\%$ and $q \geq 0.1\%$ $\Longrightarrow$ Sample size $k = 10^5$.

... usually $k \ll$ total number of active keys.

# Aggregated: ppswor estimate quality when $g() \neq f()$

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 $g, f$:**

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

# 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 $c$.

# 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* $(\frac{\rho}{q(k-1)})^{0.5}$.

# Aggregated: Proof of variance bound for sample size $k$

$$
\begin{aligned}
var[\hat{g}(w_x \mid \tau)] &= \mathsf{E}[(\hat{g}(w_x \mid \tau))^2] - g(w_x)^2 \\
&= p_{x|\tau} \frac{g(w_x)^2}{p_{x|\tau}^2} + (1 - p_{x|\tau}) \cdot 0 - g(w_x)^2 \\
&= \left( \frac{1}{p_{x|\tau}} - 1 \right) g(w_x)^2 \\
&< g(w_x)^2 \frac{e^{-\tau f(w_x)}}{1 - e^{-\tau f(w_x)}} \leq \frac{1}{\tau f(w_x)} g(w_x)^2 \leq \max_{w>0} \frac{f(w)}{g(w)} \frac{g(w_x)}{\tau} \; .
\end{aligned}
$$

# Aggregated: Proof of variance bound for sample size $k$

$$\begin{aligned}
\mathsf{var}[\hat{g}(w_x \mid \tau)] &= \mathsf{E}[(\hat{g}(w_x \mid \tau))^2] - g(w_x)^2 \\
&= p_{x|\tau}\frac{g(w_x)^2}{p_{x|\tau}^2} + (1-p_{x|\tau})\cdot 0 - g(w_x)^2 \\
&= \left(\frac{1}{p_{x|\tau}} - 1\right) g(w_x)^2 \\
&< g(w_x)^2 \frac{e^{-\tau f(w_x)}}{1-e^{-\tau f(w_x)}} \leq \frac{1}{\tau f(w_x)} g(w_x)^2 \leq \max_{w>0}\frac{f(w)}{g(w)}\frac{g(w_x)}{\tau} \enspace .
\end{aligned}$$

We take expectation over distribution of $\tau$, which is dominated by Erlang (sum of $k$ independent $\mathsf{Exp}(\sum_y f(w_y))$)

$$\mathsf{var}[\hat{g}(w_x)] \leq \mathsf{E}_{\tau\sim\mathsf{Erlang}}\mathsf{var}[\hat{g}(w_x) \mid \tau] \leq \max_{w>0}\frac{f(w)}{g(w)}\frac{g(w_x)}{(k-1)\sum_y f(w_y)} \enspace .$$

# Aggregated: Proof of variance bound for sample size $k$

$$\begin{aligned}
\mathsf{var}[\hat{g}(w_x \mid \tau)] &= \mathsf{E}[(\hat{g}(w_x \mid \tau))^2] - g(w_x)^2 \\
&= p_{x|\tau}\frac{g(w_x)^2}{p_{x|\tau}^2} + (1-p_{x|\tau})\cdot 0 - g(w_x)^2 \\
&= \left(\frac{1}{p_{x|\tau}} - 1\right) g(w_x)^2 \\
&< g(w_x)^2 \frac{e^{-\tau f(w_x)}}{1-e^{-\tau f(w_x)}} \leq \frac{1}{\tau f(w_x)} g(w_x)^2 \leq \max_{w>0}\frac{f(w)}{g(w)}\frac{g(w_x)}{\tau} \enspace .
\end{aligned}$$

We take expectation over distribution of $\tau$, which is dominated by Erlang (sum of $k$ independent $\mathsf{Exp}(\sum_y f(w_y))$)

$$\mathsf{var}[\hat{g}(w_x)] \leq \mathsf{E}_{\tau\sim\mathsf{Erlang}}\mathsf{var}[\hat{g}(w_x) \mid \tau] \leq \max_{w>0}\frac{f(w)}{g(w)}\frac{g(w_x)}{(k-1)\sum_y f(w_y)} \enspace .$$

We use zero covariances to obtain
$\mathsf{var}[\hat{Q}(g,H)] = \sum_{x\in H}\mathsf{var}[\hat{g}(w_x)] \leq \max_{w>0}\frac{f(w)}{g(w)}\frac{1}{k-1}\frac{\sum_{x\in H} g(w_x)}{\sum_x f(w_x)} \leq \frac{\rho}{q(k-1)}$

# Aggregated: Multi-Objective (MO) Samples

A weighted sample of size $k = \epsilon^{-2}$ with respect to $f$ gives estimates of $Q(g, H)$ with CV $\leq \epsilon\sqrt{\rho/q}$.
$\Longrightarrow$ guarantees on quality for $Q(g, H)$ degrades with disparity $\rho(f, g)$.

What if we want CV $\leq \epsilon/\sqrt{q}$ for several $f \in F$ ?

# Aggregated: Multi-Objective (MO) Samples

A weighted sample of size $k = \epsilon^{-2}$ with respect to $f$ gives estimates of $Q(g, H)$ with CV $\leq \epsilon\sqrt{\rho/q}$.
$\Longrightarrow$ guarantees on quality for $Q(g, H)$ degrades with disparity $\rho(f, g)$.

What if we want CV $\leq \epsilon/\sqrt{q}$ for several $f \in F$ ?
Naive solution: Use $|F|$ independent samples $S_f$ for $f \in F$. Size is $|F|\epsilon^{-2}$.

# Aggregated: Multi-Objective (MO) Samples

A weighted sample of size $k = \epsilon^{-2}$ with respect to $f$ gives estimates of $Q(g, H)$ with CV $\leq \epsilon\sqrt{\rho/q}$.
$\Longrightarrow$ guarantees on quality for $Q(g, H)$ degrades with disparity $\rho(f, g)$.

What if we want CV $\leq \epsilon/\sqrt{q}$ for several $f \in F$ ?
Naive solution: Use $|F|$ independent samples $S_f$ for $f \in F$. Size is $|F|\epsilon^{-2}$.
Can we do better ?

## Multi-objective sample $S_F$ [CKS09]

# Aggregated: Multi-Objective (MO) Samples

A weighted sample of size $k = \epsilon^{-2}$ with respect to $f$ gives estimates of $Q(g, H)$ with CV $\leq \epsilon\sqrt{\rho/q}$.
$\Longrightarrow$ guarantees on quality for $Q(g, H)$ degrades with disparity $\rho(f, g)$.

What if we want CV $\leq \epsilon/\sqrt{q}$ for several $f \in F$ ?
Naive solution: Use $|F|$ independent samples $S_f$ for $f \in F$. Size is $|F|\epsilon^{-2}$.
Can we do better ?

## Multi-objective sample $S_F$ [CKS09]

- $S_F = \bigcup_{f \in F} S_f$ is the union of *coordinated* bottom-$k$ (or pps) samples for $f \in F$: E.g. with priority sampling, draw $u_x \sim U[0, 1]$ once, and for $S_f$ use seed$(x) = u_x / f(w_x)$.
  Coordination [BEJ72, Coh97] makes similar samples $S_f$ for similar $f$.

# Aggregated: Multi-Objective (MO) Samples

A weighted sample of size $k = \epsilon^{-2}$ with respect to $f$ gives estimates of $Q(g, H)$ with CV $\leq \epsilon\sqrt{\rho/q}$.
$\Longrightarrow$ guarantees on quality for $Q(g, H)$ degrades with disparity $\rho(f, g)$.

What if we want CV $\leq \epsilon/\sqrt{q}$ for several $f \in F$ ?
Naive solution: Use $|F|$ independent samples $S_f$ for $f \in F$. Size is $|F|\epsilon^{-2}$.
Can we do better ?

## Multi-objective sample $S_F$ [CKS09]

- $S_F = \bigcup_{f \in F} S_f$ is the union of *coordinated* bottom-$k$ (or pps) samples for $f \in F$: E.g. with priority sampling, draw $u_x \sim U[0, 1]$ once, and for $S_f$ use seed$(x) = u_x / f(w_x)$.
  Coordination [BEJ72, Coh97] makes similar samples $S_f$ for similar $f$.
- For estimation, use $p_x = \Pr[x \in S_F]$ (inclusion in at least one $S_f$)

# Aggregated: Multi-Objective (MO) Samples

A weighted sample of size $k = \epsilon^{-2}$ with respect to $f$ gives estimates of $Q(g, H)$ with CV $\leq \epsilon\sqrt{\rho/q}$.
$\Longrightarrow$ guarantees on quality for $Q(g, H)$ degrades with disparity $\rho(f, g)$.

What if we want CV $\leq \epsilon/\sqrt{q}$ for several $f \in F$ ?
Naive solution: Use $|F|$ independent samples $S_f$ for $f \in F$. Size is $|F|\epsilon^{-2}$.
Can we do better ?

## Multi-objective sample $S_F$ [CKS09]

- $S_F = \bigcup_{f \in F} S_f$ is the union of *coordinated* bottom-$k$ (or pps) samples for $f \in F$: E.g. with priority sampling, draw $u_x \sim U[0, 1]$ once, and for $S_f$ use seed$(x) = u_x / f(w_x)$.
  Coordination [BEJ72, Coh97] makes similar samples $S_f$ for similar $f$.
- For estimation, use $p_x = \Pr[x \in S_F]$ (inclusion in at least one $S_f$)

- Estimates have CV $\leq \epsilon/\sqrt{q}$ for $Q(f, H)$ for all $f \in F$.
- Size *typically* $\ll |F|\epsilon^{-2}$ (but is as small as possible).

# Aggregated data: MO Sample for all monotone functions

What can we say about MO sampling the set $M$ of all monotone non-decreasing functions of $w_x$ ?

# Aggregated data: MO Sample for all monotone functions

What can we say about MO sampling the set $M$ of all monotone non-decreasing functions of $w_x$ ?

$M$ includes all moment, capping, and threshold functions ...

# Aggregated data: MO Sample for all monotone functions

What can we say about MO sampling the set $M$ of all monotone non-decreasing functions of $w_x$ ?

$M$ includes all moment, capping, and threshold functions ...

**Theorem** [Coh15b]

# Aggregated data: MO Sample for all monotone functions

What can we say about MO sampling the set $M$ of all monotone non-decreasing functions of $w_x$ ?

$M$ includes all moment, capping, and threshold functions ...

**Theorem** [Coh15b]

- Size: $\mathsf{E}[|S_M|] \leq \epsilon^{-2} \ln n$, where $n$ number of keys.

# Aggregated data: MO Sample for all monotone functions

What can we say about MO sampling the set $M$ of all monotone non-decreasing functions of $w_x$ ?

$M$ includes all moment, capping, and threshold functions ...

**Theorem** [Coh15b]

- Size: $\mathsf{E}[|S_M|] \leq \epsilon^{-2} \ln n$, where $n$ number of keys.
- Computation: $S_M$ and inclusion probabilities used for estimation can be computed using $O(n \log \epsilon^{-1})$ operations.

# Aggregated data: MO Sample for all monotone functions

What can we say about MO sampling the set $M$ of all monotone non-decreasing functions of $w_x$ ?

$M$ includes all moment, capping, and threshold functions ...

**Theorem** [Coh15b]

- Size: $\mathsf{E}[|S_M|] \leq \epsilon^{-2} \ln n$, where $n$ number of keys.
- Computation: $S_M$ and inclusion probabilities used for estimation can be computed using $O(n \log \epsilon^{-1})$ operations.
- Tight lower bound: When keys have distinct weights, any sample providing these statistical guarantees has size $\Omega(\epsilon^{-2} \ln n)$. Enough to look at thresh functions ($\mathrm{thresh}_T(x) = 1$ if $x \geq T$ and 0 otherwise)

# Aggregated data: MO Sample for all monotone functions

What can we say about MO sampling the set $M$ of all monotone non-decreasing functions of $w_x$ ?

$M$ includes all moment, capping, and threshold functions ...

**Theorem** [Coh15b]

- Size: $\mathsf{E}[|S_M|] \leq \epsilon^{-2} \ln n$, where $n$ number of keys.
- Computation: $S_M$ and inclusion probabilities used for estimation can be computed using $O(n \log \epsilon^{-1})$ operations.
- Tight lower bound: When keys have distinct weights, any sample providing these statistical guarantees has size $\Omega(\epsilon^{-2} \ln n)$. Enough to look at thresh functions ($\text{thresh}_T(x) = 1$ if $x \geq T$ and 0 otherwise)

Sampling scheme builds on a surprising relation to computing All-Distances sketches [Coh97, Coh15a])

# Summary: Aggregated data "gold standard" sampling

$\forall f() \geq 0$, with a weighted sample of size $k$ with respect to $f(w_x)$:


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

# Summary: Aggregated data "gold standard" sampling

$\forall f() \geq 0$, with a weighted sample of size $k$ with respect to $f(w_x)$:

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


With a multi-objective sample of size $\leq k \ln n$:

$\forall$ monotone $f \geq 0$, segment $H$: $\hat{Q}(f, H)$ has CV $\leq \sqrt{\frac{1}{q(f,H)k}}$.

# Summary: Aggregated data "gold standard" sampling

$\forall f() \geq 0$, with a weighted sample of size $k$ with respect to $f(w_x)$:

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


With a multi-objective sample of size $\leq k \ln n$:

$\forall$ monotone $f \geq 0$, segment $H$: $\hat{Q}(f, H)$ has CV $\leq \sqrt{\frac{1}{q(f,H)k}}$.

More properties of sampling aggregated data:

- Quality: Estimates are concentrated
- Computation: Streamed/distributed sampling with state $\propto$ sample size $k$ ! (Samples are composable)
- Applies to unaggregated data with $w_x \equiv \max_{\text{elements } (x,w)} w$

# Summary: Aggregated data "gold standard" sampling

$\forall f() \geq 0$, with a weighted sample of size $k$ with respect to $f(w_x)$:

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


With a multi-objective sample of size $\leq k \ln n$:

$\forall$ monotone $f \geq 0$, segment $H$: $\hat{Q}(f, H)$ has CV $\leq \sqrt{\frac{1}{q(f,H)k}}$.

More properties of sampling aggregated data:

- Quality: Estimates are concentrated
- Computation: Streamed/distributed sampling with state $\propto$ sample size $k$ ! (Samples are composable)
- Applies to unaggregated data with $w_x \equiv \max_{\text{elements } (x,w)} w$

Desirables with unaggregated data (and $w_x \equiv \sum_{\text{elements } (x,w)} w$):

- Computation: One or two passes, state $\propto k$ (no aggregated view!)
- Quality: Sample size/estimate quality tradeoff near gold standard.

# 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 $p \in [0, 2]$.

# 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 capping statistics.

# Sampling framework for unaggregated data [Coh15c]

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

# Sampling framework for unaggregated data [Coh15c]

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.

# Sampling framework for unaggregated data [Coh15c]

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.

# Sampling framework for unaggregated data [Coh15c]

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

## 1. Scores of elements

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

Properties of $\texttt{ElementScore}$: 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.

$$\texttt{seed}(x) = \min_{h \text{ with key } x} \texttt{ElementScore}(h)$$

# Sampling framework for unaggregated data [Coh15c]

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

## 1. Scores of elements

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

Properties of $\texttt{ElementScore}$: 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.

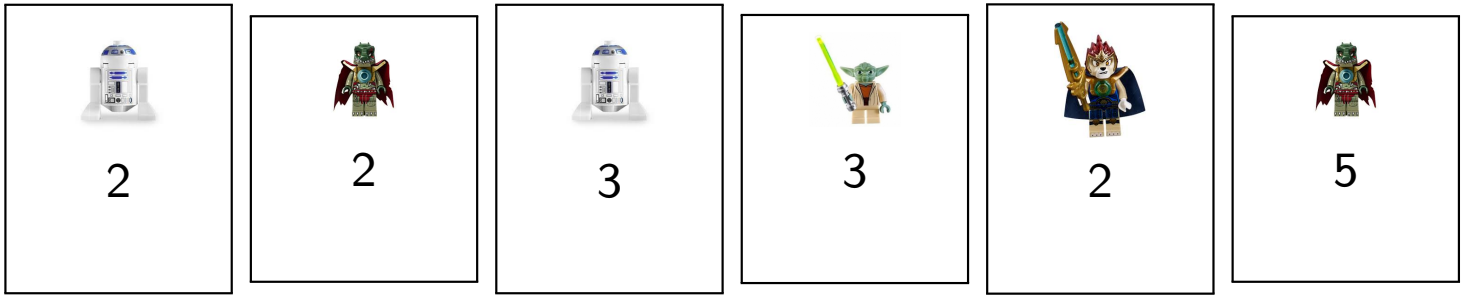
$$\texttt{seed}(x) = \min_{h \text{ with key } x} \texttt{ElementScore}(h)$$

## 3. Sample $(S, \tau)$

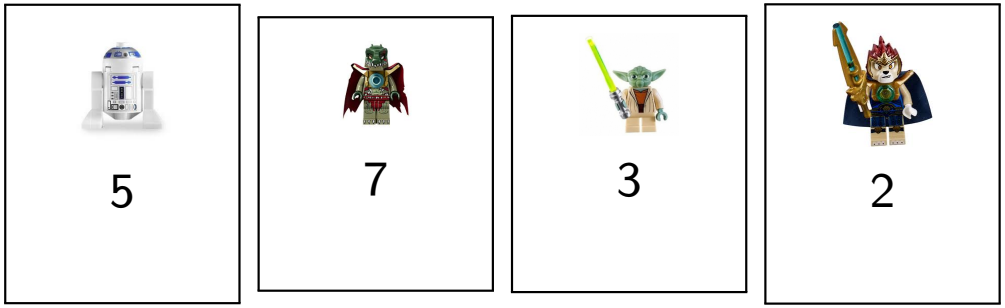
$S \leftarrow$ the $k$ keys with smallest $\texttt{seed}(x)$ (and their seed values)
$\tau \leftarrow$ the $(k+1)$st smallest seed value.

# Sampling unaggregated data: Example

Unaggregated data:

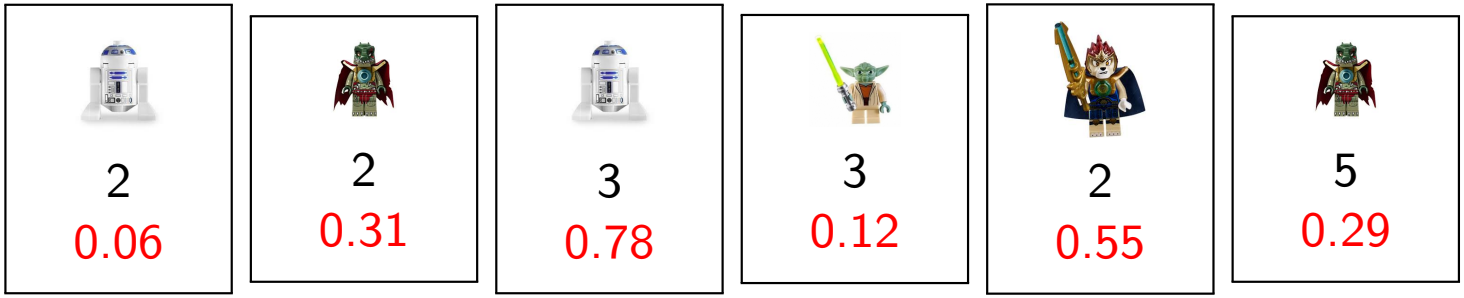



The aggregated view:

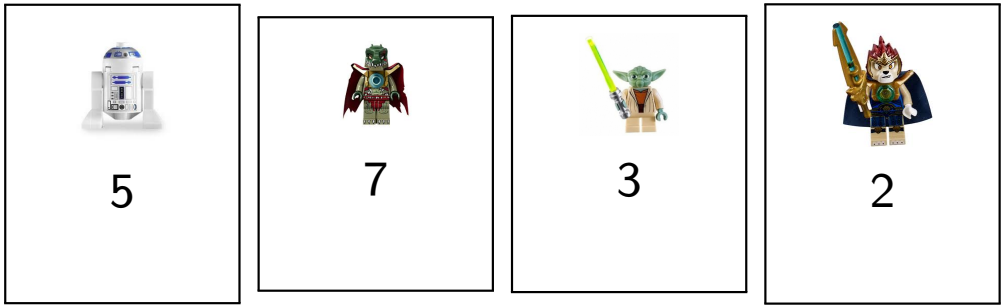


# Sampling unaggregated data: Example

Unaggregated data: (with $\texttt{ElementScore}(h)$)

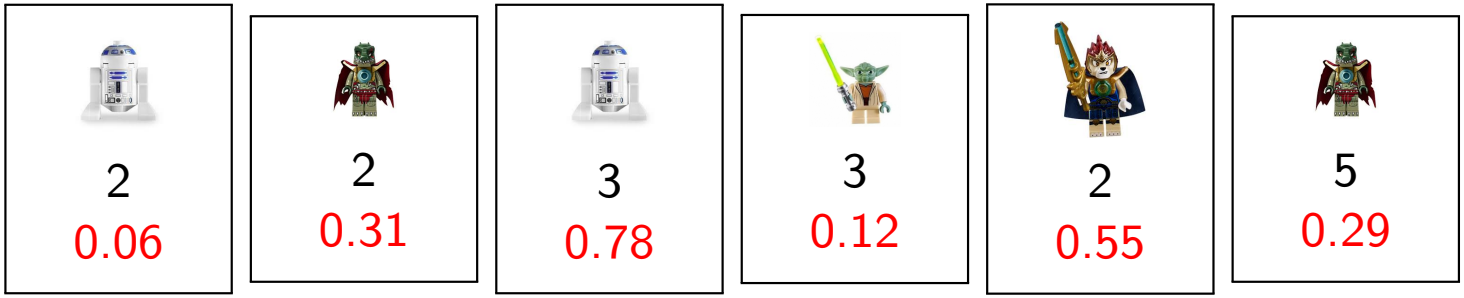



The aggregated view:

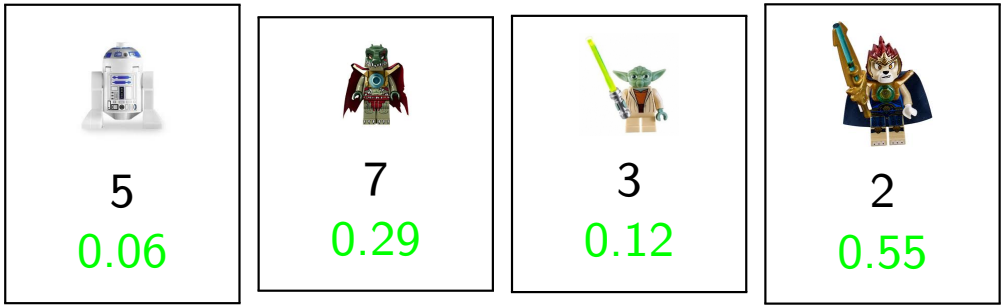


# Sampling unaggregated data: Example

Unaggregated data: (with $\texttt{ElementScore}(h)$)

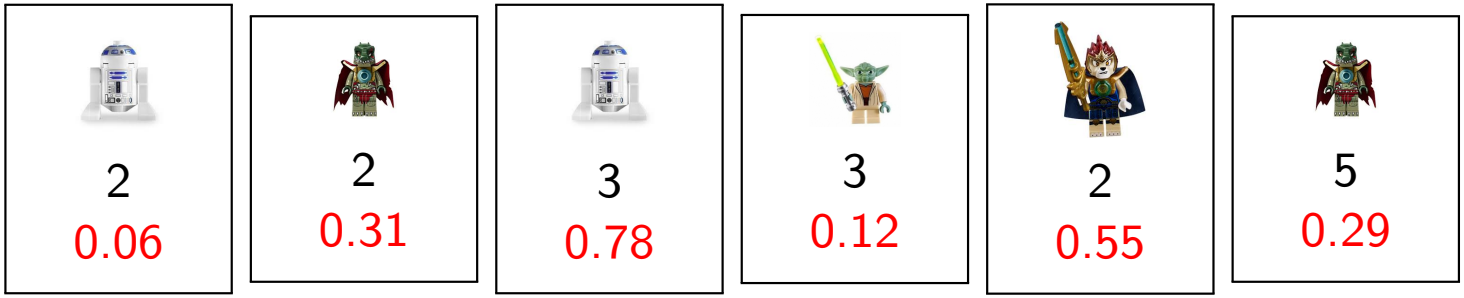



The aggregated view:
with $\texttt{seed}(x)$

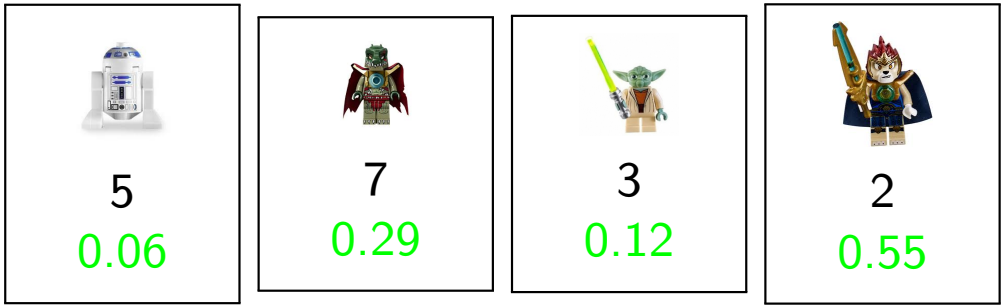


# Sampling unaggregated data: Example

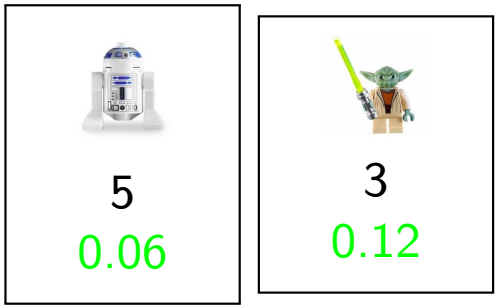
Unaggregated data: (with $\texttt{ElementScore}(h)$)




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




Sample of size $k = 2$:




$\tau = 0.29$

# Distinct sampling, casted in our framework

A distinct sample is a uniform sample of $k$ active keys (keys with $w_x > 0$). Reservoir sampling [Knu68] +Hashing [FM85] [Vit85]

# Distinct sampling, casted in our framework

A distinct sample is a uniform sample of $k$ active keys (keys with $w_x > 0$). Reservoir sampling [Knu68] +Hashing [FM85] [Vit85]

## Scoring for distinct sampling

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

# Distinct sampling, casted in our framework

A distinct sample is a uniform sample of $k$ active keys (keys with $w_x > 0$). Reservoir sampling [Knu68] +Hashing [FM85] [Vit85]

## Scoring for distinct sampling

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

Correctness: All elements with same key $x$ have the same score and thus $\texttt{seed}(x) \equiv \texttt{Hash}(x)$. The sample is the $k$ active keys with smallest hash.

# Distinct sampling, casted in our framework

A distinct sample is a uniform sample of $k$ active keys (keys with $w_x > 0$). Reservoir sampling [Knu68] +Hashing [FM85] [Vit85]

## Scoring for distinct sampling

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

Correctness: All elements with same key $x$ have the same score and thus $\texttt{seed}(x) \equiv \texttt{Hash}(x)$. The sample is the $k$ active keys with smallest hash.

From the point $x$ is included in $S$, we maintain a count $c_x$ of the sum of weights of its elements. Since any key entered the sample on its first element, we have $c_x = w_x$.

# Estimation from a distinct sample

Each key $x$ with $w_x > 0$ is sampled (conditioned on hashes of other keys) with probability $p_{x|\tau} \equiv \tau$.

# Estimation from a distinct sample

Each key $x$ with $w_x > 0$ is sampled (conditioned on hashes of other keys) with probability $p_{x|\tau} \equiv \tau$.

Since we also know $w_x$, we can use for any $f$ the unbiased inverse probability estimate [HT52]:

$$\hat{Q}(f, H) = \sum_{x \in S \cap H} \frac{f(w_x)}{p_{x|\tau}} = \frac{1}{\tau} \sum_{x \in S \cap H} f(w_x) \ .$$

# Estimation from a distinct sample

Each key $x$ with $w_x > 0$ is sampled (conditioned on hashes of other keys) with probability $p_{x|\tau} \equiv \tau$.

Since we also know $w_x$, we can use for any $f$ the unbiased inverse probability estimate [HT52]:

$$\hat{Q}(f, H) = \sum_{x \in S \cap H} \frac{f(w_x)}{p_{x|\tau}} = \frac{1}{\tau} \sum_{x \in S \cap H} f(w_x) \ .$$

Estimate quality: The sample and estimator are ppswor with respect to weights $f(w) \equiv \mathrm{cap}_1$

$\Longrightarrow$ For a segment $H$ with proportion $q$, $\hat{Q}(\mathrm{cap}_1, H)$ has CV $\approx \sqrt{\frac{1}{qk}}$.

# Estimation from a distinct sample

Each key $x$ with $w_x > 0$ is sampled (conditioned on hashes of other keys) with probability $p_{x|\tau} \equiv \tau$.

Since we also know $w_x$, we can use for any $f$ the unbiased inverse probability estimate [HT52]:

$$\hat{Q}(f, H) = \sum_{x \in S \cap H} \frac{f(w_x)}{p_{x|\tau}} = \frac{1}{\tau} \sum_{x \in S \cap H} f(w_x) \ .$$

Estimate quality: The sample and estimator are ppswor with respect to weights $f(w) \equiv \mathrm{cap}_1$

$\Longrightarrow$ For a segment $H$ with proportion $q$, $\hat{Q}(\mathrm{cap}_1, H)$ has CV $\approx \sqrt{\frac{1}{qk}}$.

$\Longrightarrow$ For $\mathrm{cap}_T$ statistics, disparity is $\rho(\mathrm{cap}_1, \mathrm{cap}_T) = T$. The bound on the CV of $\hat{Q}(\mathrm{cap}_T, H)$ is $\sqrt{\frac{T}{qk}}$. Intuitively, our sample can easily miss "heavy" keys with high $\mathrm{cap}_T(w_x)$ values which contribute more to the statistics.

# Sampling for sum statistics

Sample and Hold (counting samples) [GM98, EV02]:

If $x \in S$, increment $c_x$. Otherwise, cache if $\texttt{rand}() < \tau$.

Can be used with a fixed-size sample $k$; Equivalent to ppswor [CDK+14]; Continuous version (element weights) [CCD11].

# Sampling for sum statistics

Sample and Hold (counting samples) [GM98, EV02]:

If $x \in S$, increment $c_x$. Otherwise, cache if $\texttt{rand()} < \tau$.

Can be used with a fixed-size sample $k$; Equivalent to ppswor [CDK+14]; Continuous version (element weights) [CCD11].

Sample and Hold casted in our framework:

**Element scoring function**

$$\texttt{ElementScore}(h{=}(x,w)) \sim \mathrm{Exp}[w]$$

# Sampling for sum statistics

Sample and Hold (counting samples) [GM98, EV02]:

If $x \in S$, increment $c_x$. Otherwise, cache if $\texttt{rand()} < \tau$.

Can be used with a fixed-size sample $k$; Equivalent to ppswor [CDK+14]; Continuous version (element weights) [CCD11].

Sample and Hold casted in our framework:

**Element scoring function**

$$\texttt{ElementScore}(h{=}(x,w)) \sim \mathrm{Exp}[w]$$

The minimum of independent exponential random variables is an exponential random variable with a parameter that is the sum of their parameters. We get

$$\mathrm{seed}(x) \sim \min_{\text{elements } (x,w)} \mathrm{Exp}[w] \equiv \mathrm{Exp}[w_x] \implies \text{ppswor wrt } w_x!$$

# Unaggregated data: Estimating sum statistics from ppswor

Caveat! We do have a ppswor sample $S$ and the threshold $\tau$, but **exact** weights $w_x$ for $x \in S$ are needed for the inverse probability estimator. When streaming (single pass), we can start "counting" $w_x$ only after $x$ enters the cache, so we may miss some elements and only have $c_x < w_x$.

# Unaggregated data: Estimating sum statistics from ppswor

Caveat! We do have a ppswor sample $S$ and the threshold $\tau$, but **exact** weights $w_x$ for $x \in S$ are needed for the inverse probability estimator. When streaming (single pass), we can start "counting" $w_x$ only after $x$ enters the cache, so we may miss some elements and only have $c_x < w_x$.

Solutions:

# Unaggregated data: Estimating sum statistics from ppswor

Caveat! We do have a ppswor sample $S$ and the threshold $\tau$, but **exact** weights $w_x$ for $x \in S$ are needed for the inverse probability estimator. When streaming (single pass), we can start "counting" $w_x$ only after $x$ enters the cache, so we may miss some elements and only have $c_x < w_x$.

Solutions:

- 2-passes: Use the first pass to identify the set $S$ of sampled keys. Use a second pass to exactly count $w_x$ for sampled keys. Apply ppswor inverse probability estimator.

# Unaggregated data: Estimating sum statistics from ppswor

Caveat! We do have a ppswor sample $S$ and the threshold $\tau$, but **exact** weights $w_x$ for $x \in S$ are needed for the inverse probability estimator. When streaming (single pass), we can start "counting" $w_x$ only after $x$ enters the cache, so we may miss some elements and only have $c_x < w_x$.

Solutions:

- 2-passes: Use the first pass to identify the set $S$ of sampled keys. Use a second pass to exactly count $w_x$ for sampled keys. Apply ppswor inverse probability estimator.
- Work with $c_x$: For estimating sum statistics, we can add expected weight of missed prefix [GM98, EV02, CDK+14] (discrete) [CCD11] (continuous) to each sampled key in segment to obtain an unbiased estimate.

Possible to estimate unbiasedly general $f$.... [CDK+14] (discrete) [Coh15c] (continuous)... more later.

# $\ell$-capped sampling [Coh15c]

**Hurdle 1**

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

# $\ell$-capped sampling [Coh15c]

## Hurdle 1

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

## Hurdle 2

Streaming: Even if we have the "right" sampling probabilities, when using a single pass we need estimators that work with observed counts $c_x$ instead of with $w_x$

# $\ell$-capped sampling: Hurdle 1 

Obtaining inclusion probabilities roughly proportional to $\mathrm{cap}_\ell(w_x)$

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

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

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

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

# $\ell$-capped sampling: Hurdle 1 

Obtaining inclusion probabilities roughly proportional to $\mathrm{cap}_\ell(w_x)$

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

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

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

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

**$\texttt{seed}(x)$ distribution**

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

# $\ell$-capped sampling: Hurdle 1 

Obtaining inclusion probabilities roughly proportional to $\mathrm{cap}_\ell(w_x)$

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

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

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

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

**$\texttt{seed}(x)$ distribution**

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

- For keys with $w_x \ll \ell$, this is like ppswor wrt $w_x$

# $\ell$-capped sampling: Hurdle 1 

Obtaining inclusion probabilities roughly proportional to $\mathrm{cap}_\ell(w_x)$

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

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

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

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

**$\texttt{seed}(x)$ distribution**

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

- 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

With 2-passes, we have $w_x$, can compute inclusion probabilities from $\tau$ and the distribution, and 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*

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

# 2-pass estimation quality

With 2-passes, we have $w_x$, can compute inclusion probabilities from $\tau$ and the distribution, and 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*

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

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

# 2-pass estimation quality

With 2-passes, we have $w_x$, can compute inclusion probabilities from $\tau$ and the distribution, and 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*

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

- $\rho = \max\{T/\ell, \ell/T\}$ is the disparity between $\text{cap}_\ell$ and $\text{cap}_T$.
- Overhead factor of $(\frac{e}{e-1})^{0.5} \approx 1.26$ over aggregated "gold standard."

# 2-pass estimation quality

With 2-passes, we have $w_x$, can compute inclusion probabilities from $\tau$ and the distribution, and 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*

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

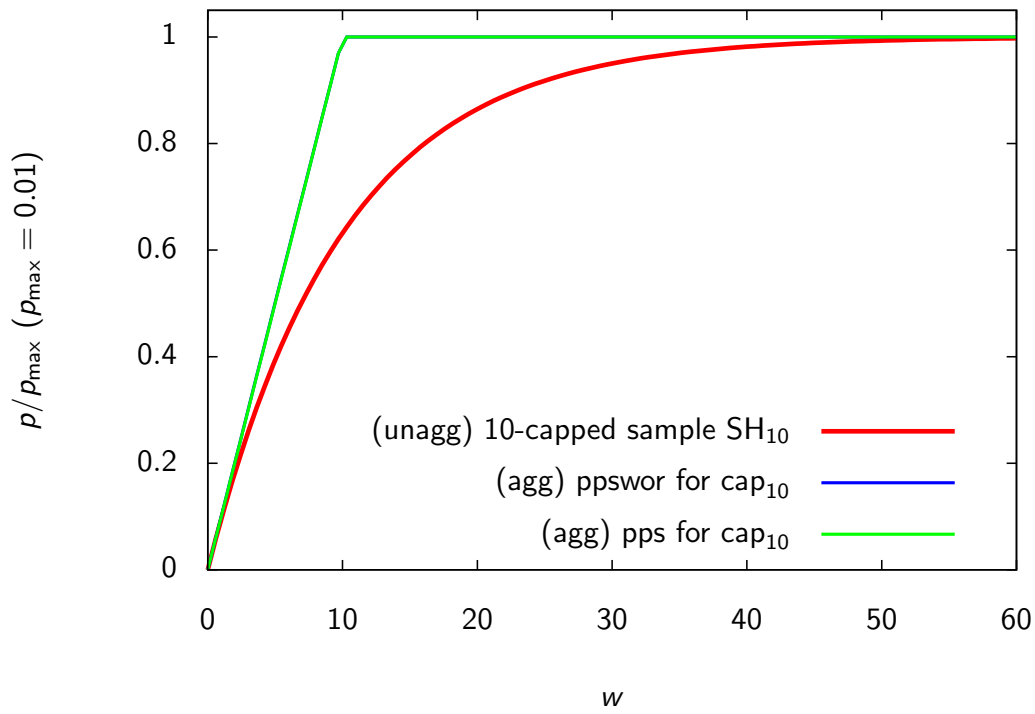
- $\rho = \max\{T/\ell, \ell/T\}$ is the disparity between $\text{cap}_\ell$ and $\text{cap}_T$.
- Overhead factor of $(\frac{e}{e-1})^{0.5} \approx 1.26$ over aggregated "gold standard."
- This is a worst case factor (many items with $w_x = O(\ell)$)

# Estimation quality: 2-pass vs. gold standard

10-capped sample, pps and ppswor with weights $\mathrm{cap}_{10}(w)$.

- $x$ axis: the key weight $w$
- $y$ axis: ratio of inclusion probability to max inclusion probability (set to 0.01).

Ratio gap between curves is maximizes at $w = 10$ and is $(1 - 1/e)$. It is the loss of 10-capped versus aggregated gold standard.



# Streaming estimators: Hurdle 2 

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

# 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 $c_x \leftarrow c_x + w$.

# 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.

# 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.

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

# 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.

$\Longrightarrow$ $c_x$ 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 $Y^{-1}$:

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

# 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.

$\Longrightarrow$ $c_x$ 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

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

# 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.

$\Longrightarrow$ $c_x$ 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 $\hat{Q}(cap_T, H)$ applied to an $\ell$-capped sample is upper bounded by*

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

# Streaming estimator quality

**Theorem**

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

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

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{Sample[x].seed} then
            Initialize Sample[x]
            Sample[x].seed ← s;
            if |Sample| = k + 1 then
                y ← arg max{Sample[x].seed}
                delete Sample[y]

// Pass I: Merge two summaries Sample, Sample2
foreach x ∈ 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 |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 ∈ Sample do // Initialize thread
    Sample[x].w ← 0

foreach element h = (x, w) do
    if x ∈ Sample then
        Sample[x].w ← Sample[x] + w

// Pass II: Merge two summaries Sample, Sample2
foreach x ∈ Sample do
    Sample[x].w ← Sample[x].w + Sample2[x].w
```

# (pseudo) Code: Fixed-$k$ stream $\ell$-capped sampling

**foreach** *stream element* $(x, w)$ **do** // Process element

  **if** $x$ *is in* Counters **then**

    $\text{Counters}[x] \leftarrow \text{Counters}[x] + w$;

  **else**

    $\Delta \leftarrow -\frac{\ln(1-\texttt{rand}())}{\max\{\ell^{-1}, \tau\}}$ // $\sim \text{Exp}[\max\{\ell^{-1}, \tau\}]$

    **if** $\Delta < w$ *and* $(\tau\ell > 1$ *or* $\tau\ell \leq 1$ *and* $\texttt{KeyBase}(x) < \tau)$ **then** // insert $x$

      $\text{Counters}[x] \leftarrow w - \Delta$

      **if** $|\text{Counters}| = k + 1$ **then** // Evict a key

        **if** $\tau\ell > 1$ **then**

          **foreach** $x \in$ Counters **do**

            $u_x \leftarrow \texttt{rand}()$; $r_x \leftarrow \texttt{rand}()$ ; $z_x \leftarrow \min\{\tau u_x, \frac{-\ln(1-r_x)}{\text{Counters}[x]}\}$ // $x$'s evict threshold

            **if** $z_x \leq \ell^{-1}$ **then**

              $z_x \leftarrow \texttt{KeyBase}(x)$

          $y \leftarrow \arg\max_{x \in \text{Counters}} z_x$; delete $y$ from Counters // key to evict

          $\tau^* \leftarrow z_y$ // new threshold

          **foreach** $x \in$ Counters **do** // Adjust counters according to $\tau^*$

            **if** $u_x > \max\{\tau^*, \ell^{-1}\}/\tau$ **then**

              $\text{Counters}[x] \overline{\leftarrow} \frac{-\ln(1-r_x)}{\max\{\ell^{-1}, \tau^*\}}$

          $\tau \leftarrow \tau^*$; delete $u, r, z, b$ // deallocate memory

        **else** // $\tau\ell \leq 1$

          $y \leftarrow \arg\max_{x \in \text{Counters}} \texttt{KeyBase}(x)$; Delete $y$ from Counters // evict $y$

          $\tau \leftarrow \texttt{KeyBase}(y)$ // new threshold

**return**$(\tau;\ (x, \text{Counters}[x])$ *for* $x$ *in* Counters$)$

# Simulations

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

What is the behavior on realistic instances ?

- Quantify gain from second pass
- Understand actual dependence on disparity
- How much do we 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 $Q(\mathrm{cap}_T, \mathcal{X})$ from $\ell$-capped sample.

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

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

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

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

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 is lower than worst-case:
  - We do not see the $\sqrt{e/(e-1)}$ factor (comes in when many keys have $w_x \approx \ell$).
  - Gain from skew: Observed for large $T$
    - Note that when $T \ll \ell$, skew can hurt us on "worst-case" segments of many light keys

# Observations from Simulations

- Actual NRMSE is lower than worst-case:
  - We do not see the $\sqrt{e/(e-1)}$ factor (comes in when many keys have $w_x \approx \ell$).
  - Gain from skew: Observed for large $T$
    - Note that when $T \ll \ell$, skew can hurt us on "worst-case" segments of many light keys
- Much better to use $\ell \approx T$

# Observations from Simulations

- Actual NRMSE is lower than worst-case:
  - We do not see the $\sqrt{e/(e-1)}$ factor (comes in when many keys have $w_x \approx \ell$).
  - Gain from skew: Observed for large $T$
    - Note that when $T \ll \ell$, skew can hurt us on "worst-case" segments of many light keys
- Much better to use $\ell \approx T$
- 2-pass estimation quality is within 10% of 1-pass ( $\Longrightarrow$ use 2-pass to distribute computation but not to improve estimation)

# Conclusion

Summary:

- Aggregated data: Optimal multi-objective sampling scheme for all monotone $f$

# Conclusion

Summary:

- Aggregated data: Optimal multi-objective sampling scheme for all monotone $f$
- Unaggregated data: Sampling framework which unifies and extends classic solutions for distinct and sum statistics.
- First solution for mid-range $\mathrm{cap}_T$ statistics, nearly matches aggregated gold standard.

# Conclusion

Summary:

- Aggregated data: Optimal multi-objective sampling scheme for all monotone $f$
- Unaggregated data: Sampling framework which unifies and extends classic solutions for distinct and sum statistics.
- First solution for mid-range $\mathrm{cap}_T$ statistics, nearly matches aggregated gold standard.

Natural Questions (with partial answers):

- Which other monotone frequency functions can our framework handle, in near "aggregated gold standard" sense?

# Conclusion

Summary:

- Aggregated data: Optimal multi-objective sampling scheme for all monotone $f$
- Unaggregated data: Sampling framework which unifies and extends classic solutions for distinct and sum statistics.
- First solution for mid-range $\mathrm{cap}_T$ statistics, nearly matches aggregated gold standard.

Natural Questions (with partial answers):

- Which other monotone frequency functions can our framework handle, in near "aggregated gold standard" sense?
  - Some functions are "hard" for streaming (polynomial lower bounds on state): E.g., moments with $p > 2$ [AMS99], threshold

# Conclusion

Summary:

- Aggregated data: Optimal multi-objective sampling scheme for all monotone $f$
- Unaggregated data: Sampling framework which unifies and extends classic solutions for distinct and sum statistics.
- First solution for mid-range $\mathrm{cap}_T$ statistics, nearly matches aggregated gold standard.

Natural Questions (with partial answers):

- Which other monotone frequency functions can our framework handle, in near "aggregated gold standard" sense?
  - Some functions are "hard" for streaming (polynomial lower bounds on state): E.g., moments with $p > 2$ [AMS99], threshold
  - Can handle any $f$ that is a nonnegative combination of $\mathrm{cap}_T$ functions: All $f$ such that $f' \leq 1$ and $f'' \leq 0$.

# Conclusion

Summary:

- Aggregated data: Optimal multi-objective sampling scheme for all monotone $f$
- Unaggregated data: Sampling framework which unifies and extends classic solutions for distinct and sum statistics.
- First solution for mid-range $\mathrm{cap}_T$ statistics, nearly matches aggregated gold standard.

Natural Questions (with partial answers):

- Which other monotone frequency functions can our framework handle, in near "aggregated gold standard" sense?
  - Some functions are "hard" for streaming (polynomial lower bounds on state): E.g., moments with $p > 2$ [AMS99], threshold
  - Can handle any $f$ that is a nonnegative combination of $\mathrm{cap}_T$ functions: All $f$ such that $f' \leq 1$ and $f'' \leq 0$.
  - Can also obtain a multi-objective sample for these functions (logarithmic factor on sample size)

# Conclusion

Summary:

- Aggregated data: Optimal multi-objective sampling scheme for all monotone $f$
- Unaggregated data: Sampling framework which unifies and extends classic solutions for distinct and sum statistics.
- First solution for mid-range $\mathrm{cap}_T$ statistics, nearly matches aggregated gold standard.

Natural Questions (with partial answers):

- Which other monotone frequency functions can our framework handle, in near "aggregated gold standard" sense?
  - Some functions are "hard" for streaming (polynomial lower bounds on state): E.g., moments with $p > 2$ [AMS99], threshold
  - Can handle any $f$ that is a nonnegative combination of $\mathrm{cap}_T$ functions: All $f$ such that $f' \leq 1$ and $f'' \leq 0$.
  - Can also obtain a multi-objective sample for these functions (logarithmic factor on sample size)
  - What about $f$ with super-linear growth? say $p \in (1, 2]$ moments (handled by linear sketches+stable distributions [Ind01, MW10])
  - Can we support signed updates where $f(\max\{0, w\})$? Perhaps build on techniques from [GLH06, CCD12, Coh15c].

# Conclusion

Summary:

- Aggregated data: Optimal multi-objective sampling scheme for all monotone $f$
- Unaggregated data: Sampling framework which unifies and extends classic solutions for distinct and sum statistics.
- First solution for mid-range $\mathrm{cap}_T$ statistics, nearly matches aggregated gold standard.

Natural Questions (with partial answers):

- Which other monotone frequency functions can our framework handle, in near "aggregated gold standard" sense?
- Can we do other aggregates of the elements of a given key ?

# Conclusion

Summary:

- Aggregated data: Optimal multi-objective sampling scheme for all monotone $f$
- Unaggregated data: Sampling framework which unifies and extends classic solutions for distinct and sum statistics.
- First solution for mid-range $\mathrm{cap}_T$ statistics, nearly matches aggregated gold standard.

Natural Questions (with partial answers):

- Which other monotone frequency functions can our framework handle, in near "aggregated gold standard" sense?
- Can we do other aggregates of the elements of a given key ?
  - (functions of) Sum: here
  - (functions of) max: small extension to aggregated sampling (through sample coordination)
  - what other aggregations are interesting and can be handled ?

# Conclusion

Summary:

- Aggregated data: Optimal multi-objective sampling scheme for all monotone $f$
- Unaggregated data: Sampling framework which unifies and extends classic solutions for distinct and sum statistics.
- First solution for mid-range $\mathrm{cap}_T$ statistics, nearly matches aggregated gold standard.

Natural Questions (with partial answers):

- Which other monotone frequency functions can our framework handle, in near "aggregated gold standard" sense?
- Can we do other aggregates of the elements of a given key ?
- If we only want $Q(\mathrm{cap}_T, \mathcal{X})$, can we do better ?
  - Is there a "Hyperloglog like" [FFGM07] algorithm with sketch size $O(\epsilon^{-2} + \log\log n)$ (instead of $O(\epsilon^{-2} \log n)$) ?
  - Can we use HIP estimators? [Coh15a, Tin14]

Thank you!

# Bibliography I


N. Alon, Y. Matias, and M. Szegedy.
The space complexity of approximating the frequency moments.
*J. Comput. System Sci.*, 58:137–147, 1999.

K. R. W. Brewer, L. J. Early, and S. F. Joyce.
Selecting several samples from a single population.
*Australian Journal of Statistics*, 14(3):231–239, 1972.

E. Cohen, G. Cormode, and N. Duffield.
Structure-aware sampling: Flexible and accurate summarization.
*Proceedings of the VLDB Endowment*, 2011.

E. Cohen, G. Cormode, and N. Duffield.
Don't let the negatives bring you down: Sampling from streams of signed updates.
In *Proc. ACM SIGMETRICS/Performance*, 2012.

E. Cohen, N. Duffield, H. Kaplan, C. Lund, and M. Thorup.
Algorithms and estimators for accurate summarization of unaggregated data streams.
*J. Comput. System Sci.*, 80, 2014.

E. Cohen, N. Duffield, C. Lund, M. Thorup, and H. Kaplan.
Efficient stream sampling for variance-optimal estimation of subset sums.
*SIAM J. Comput.*, 40(5), 2011.

M. T. Chao.
A general purpose unequal probability sampling plan.
*Biometrika*, 69(3):653–656, 1982.

# Bibliography II



E. Cohen and H. Kaplan.
Summarizing data using bottom-k sketches.
In *ACM PODC*, 2007.


E. Cohen, H. Kaplan, and S. Sen.
Coordinated weighted sampling for estimating aggregates over multiple weight assignments.
*VLDB*, 2(1–2), 2009.
full: `http://arxiv.org/abs/0906.4560`.


E. Cohen.
Size-estimation framework with applications to transitive closure and reachability.
*J. Comput. System Sci.*, 55:441–453, 1997.


E. Cohen.
All-distances sketches, revisited: HIP estimators for massive graphs analysis.
*TKDE*, 2015.


E. Cohen.
Multi-objective weighted sampling.
In *HotWeb*. IEEE, 2015.
full version: `http://arxiv.org/abs/1509.07445`.


E. Cohen.
Stream sampling for frequency cap statistics.
In *KDD*. ACM, 2015.
full version: `http://arxiv.org/abs/1502.05955`.

# Bibliography III


N. Duffield, M. Thorup, and C. Lund.

Priority sampling for estimating arbitrary subset sums.

*J. Assoc. Comput. Mach.*, 54(6), 2007.

C. Estan and G. Varghese.

New directions in traffic measurement and accounting.

In *SIGCOMM*. ACM, 2002.

P. Flajolet, E. Fusy, O. Gandouet, and F. Meunier.

Hyperloglog: The analysis of a near-optimal cardinality estimation algorithm.

In *Analysis of Algorithms (AofA)*. DMTCS, 2007.

P. Flajolet and G. N. Martin.

Probabilistic counting algorithms for data base applications.

*J. Comput. System Sci.*, 31:182–209, 1985.

R. Gemulla, W. Lehner, and P. J. Haas.

A dip in the reservoir: Maintaining sample synopses of evolving datasets.

In *VLDB*, 2006.

P. Gibbons and Y. Matias.

New sampling-based summary statistics for improving approximate query answers.

In *SIGMOD*. ACM, 1998.

D. G. Horvitz and D. J. Thompson.

A generalization of sampling without replacement from a finite universe.

*Journal of the American Statistical Association*, 47(260):663–685, 1952.

# Bibliography IV


P. Indyk.

Stable distributions, pseudorandom generators, embeddings and data stream computation.
In *Proc. 41st IEEE Annual Symposium on Foundations of Computer Science*, pages 189–197. IEEE, 2001.

W. Johnson and J. Lindenstrauss.

Extensions of Lipschitz mappings into a Hilbert space.
*Contemporary Math.*, 26, 1984.

D. E. Knuth.

*The Art of Computer Programming, Vol 2, Seminumerical Algorithms*.
Addison-Wesley, 1st edition, 1968.

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.
Technical report, Cornell University, 1982.

M. Monemizadeh and D. P. Woodruff.

1-pass relative-error $l_p$-sampling with applications.
In *Proc. 21st ACM-SIAM Symposium on Discrete Algorithms*. ACM-SIAM, 2010.

# Bibliography V


E. Ohlsson.

Sequential poisson sampling.

*J. Official Statistics*, 14(2):149–162, 1998.

B. Rosén.

Asymptotic theory for successive sampling with varying probabilities without replacement, I.

*The Annals of Mathematical Statistics*, 43(2):373–397, 1972.

B. Rosén.

Asymptotic theory for order sampling.

*J. Statistical Planning and Inference*, 62(2):135–158, 1997.

D. Ting.

Streamed approximate counting of distinct elements: Beating optimal batch methods.

In *KDD*. ACM, 2014.

J.S. Vitter.

Random sampling with a reservoir.

*ACM Trans. Math. Softw.*, 11(1):37–57, 1985.