Stream Sampling for Frequency Cap Statistics

Edith Cohen

[1]Google, CA USA

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

August 11, 2015

Model: Aggregated / Unaggregated Data

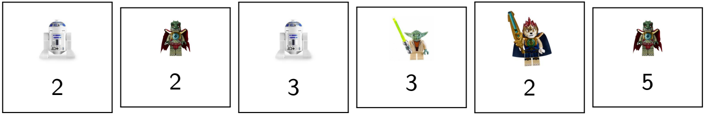
Data *elements* (x, w) have a *key* x and a numeric *value* $w > 0$

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

Model: Aggregated / Unaggregated Data

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

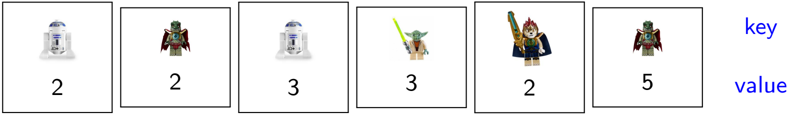
- Elements are streamed or distributed, no particular order/partition
- "Unaggregated:" Multiple elements can have the same key
- "Aggregated:" Elements have unique keys



Model: Aggregated / Unaggregated Data

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

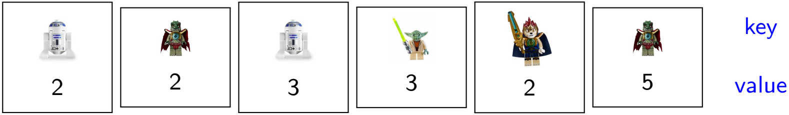
- Elements are streamed or distributed, no particular order/partition
- "Unaggregated:" Multiple elements can have the same key
- "Aggregated:" Elements have unique keys



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

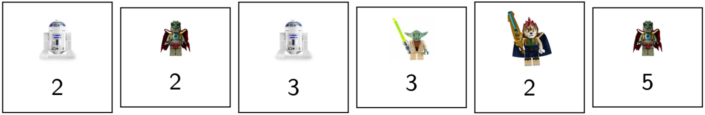


The *aggregated view*: The set of key value pairs (x, w_x) for active keys x. w_x is the sum of values of elements with key x.

Model: Aggregated / Unaggregated Data

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

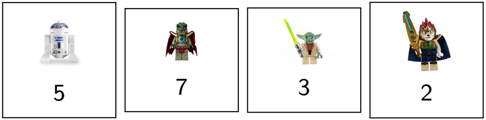
- Elements are streamed or distributed, no particular order/partition
- "Unaggregated:" Multiple elements can have the same key
- "Aggregated:" Elements have unique keys



key

value

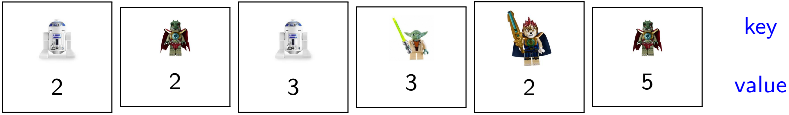
The *aggregated view*: The set of key value pairs (x, w_x) for active keys x. w_x is the sum of values of elements with key x.



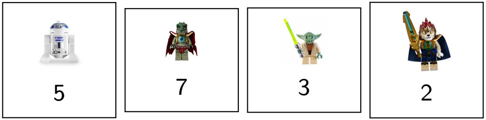
Model: Aggregated / Unaggregated Data

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 x. w_x is the sum of values of elements with key x.



Queries are often specified over the aggregated view

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.

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 $f()$:

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 $p = 1$)
- 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 $f(w) = w^p$ $(p \geq 0)$ (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 cap_T with $T \in (0, +\infty)$ parametrize the range between distinct and sum.

Use case: Frequency capping in online advertising

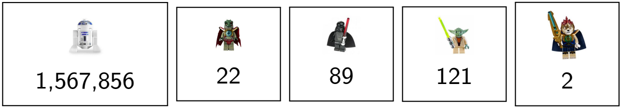
The first few impressions of the same ad per user are more effective than later ones (diminishing return). Advertisers therefore specify

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

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.



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

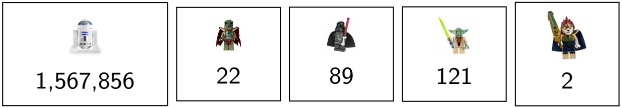

2

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

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.



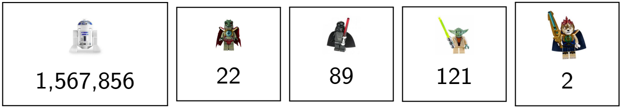
Q: targeted segment: galactic-scale travelers cap: 5

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.



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

... Frequency Capping in Online advertising

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.

... Frequency Capping in Online advertising

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 $Q(\text{cap}_T, H)$ 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 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 $Q(\text{cap}_T, H)$ of qualifying impressions.

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

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

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

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

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:

- *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, $H \subseteq \mathcal{X}$
- 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) $\Pr[x \in S] \propto f(w_x)$:

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

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$(x) \sim Z[f(w_x)]$

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

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**
  $\texttt{seed}(x) \sim Z[f(w_x)]$
$S \leftarrow k$ keys with smallest $\texttt{seed}(x)$; $\tau \leftarrow (k+1)$th smallest $\texttt{seed}(x)$

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

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 $x \in S$.

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 $g(w_x) > 0 \implies f(w_x) > 0$)

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

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

- 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 $x \notin S$ 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

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

- 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?

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)

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

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)

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

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

Aggregated: 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 $(\frac{\rho}{q(k-1)})^{0.5}$.

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 $\leq \sqrt{\frac{\rho(g,f)}{q(g,H)k}}$

Now back to unaggregated data. Desirables:

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 $\leq \sqrt{\frac{\rho(g,f)}{q(g,H)k}}$

Now back to unaggregated data. Desirables:

- Computation: One pass (when streaming) or two passes with state proportional to sample size k ($\Longrightarrow$ can't compute aggregated view)
- Estimation: Sample size/estimation quality tradeoff close to gold standard.

Toolbox for frequency functions on unaggregated streams

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 cap statistics.

Sampling framework for unaggregated data

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

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 $\texttt{ElementScore}(h)$ 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 `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

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

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 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.

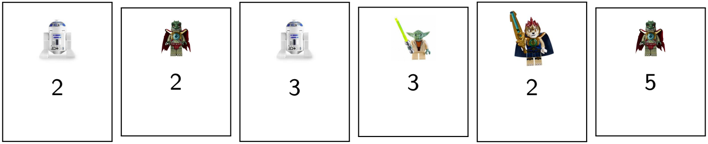
$$\texttt{seed}(x) = \min_{h \text{ with key } x} \texttt{ElementScore}(h)$$

3. Sample (S, τ)

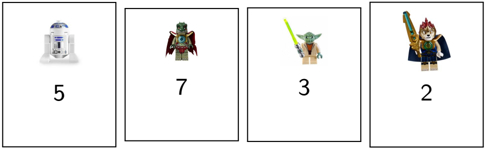
$S \leftarrow$ the k keys with smallest $\texttt{seed}(x)$ (and their seed values)
$\tau \leftarrow$ the $(k+1)$st smallest seed value.

Sampling unaggregated data: Example

Unaggregated data:



The aggregated view:



Sampling unaggregated data: Example

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


2
0.06


2
0.31


3
0.78


3
0.12


2
0.55


5
0.29

The aggregated view:


5


7


3


2

Sampling unaggregated data: Example

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

					
2	2	3	3	2	5
0.06	0.31	0.78	0.12	0.55	0.29

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

			
5	7	3	2
0.06	0.29	0.12	0.55

Sampling unaggregated data: Example

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


2
0.06


2
0.31


3
0.78


3
0.12


2
0.55


5
0.29

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


5
0.06


7
0.29


3
0.12


2
0.55

Sample of size $k = 2$:


5
0.06


3
0.12

$\tau = 0.29$

Unaggregated Sampling for distinct and sum

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

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

Unaggregated Sampling for distinct and sum

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

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

Correctness: We have $\texttt{seed}(x) \equiv \texttt{Hash}(x) \implies$ 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])

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

Unaggregated Sampling for distinct and sum

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

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

Correctness: We have $\texttt{seed}(x) \equiv \texttt{Hash}(x) \implies$ 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])

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

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

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

Sampling for cap statistics: ℓ-capped sampling

Hurdle 1

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

Sampling for cap statistics: ℓ-capped sampling

Hurdle 1

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

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

ℓ-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 ℓ-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 $\texttt{KeyBase}(x)$.

ℓ-capped sampling: Hurdle 1 

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

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 $v \sim \text{Exp}[w]$ and then returning v if $v > 1/\ell$ and $\texttt{KeyBase}(x)$ otherwise:

element scoring for ℓ-capped samples

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

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

$\texttt{seed}(x)$ distribution

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

ℓ-capped sampling: Hurdle 1 

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

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 $v \sim \text{Exp}[w]$ and then returning v if $v > 1/\ell$ and $\texttt{KeyBase}(x)$ otherwise:

element scoring for ℓ-capped samples

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

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

$\texttt{seed}(x)$ distribution

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

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

ℓ-capped sampling: Hurdle 1 

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

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 $v \sim \text{Exp}[w]$ and then returning v if $v > 1/\ell$ and $\texttt{KeyBase}(x)$ otherwise:

element scoring for ℓ-capped samples

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

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

$\texttt{seed}(x)$ distribution

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

- 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

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 ℓ-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

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 ℓ-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 cap_ℓ and cap_T.

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 ℓ-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 cap_ℓ and cap_T.
- Overhead factor of $(\frac{e}{e-1})^{0.5} \approx 1.26$ over 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 τ 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 τ 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 τ 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}:

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

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 τ 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 τ 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 $\hat{Q}(cap_T, H)$ applied to an ℓ-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 ℓ-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 ℓ-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 ℓ-capped sampling

```
foreach stream element (x, w) do // Process element
    if x is in Counters then
        Counters[x] ← Counters[x] + w;
    else
        Δ ← − ln(1−rand()) / max{ℓ^−1, τ} // ∼ Exp[max{ℓ^−1, τ}]
        if Δ < w and (τℓ > 1 or τℓ ≤ 1 and KeyBase(x) < τ) then // insert x
            Counters[x] ← w − Δ
            if |Counters| = k + 1 then // Evict a key
                if τℓ > 1 then
                    foreach x ∈ Counters do
                        u_x ← rand(); r_x ← rand() ; z_x ← min{τu_x, −ln(1−r_x)/Counters[x]}// x's evict threshold
                        if z_x ≤ ℓ^−1 then
                            z_x ← KeyBase(x)
                    y ← arg max_{x∈Counters} z_x; delete y from Counters // key to evict
                    τ* ← z_y// new threshold
                    foreach x ∈ Counters do // Adjust counters according to τ*
                        if u_x > max{τ*, ℓ^−1}/τ then
                            Counters[x] ←(−) −ln(1−r_x) / max{ℓ^−1, τ*}
                    τ ← τ*; delete u, r, z, b // deallocate memory
                else // τℓ ≤ 1
                    y ← arg max_{x∈Counters} KeyBase(x); Delete y from Counters // evict y
                    τ ← KeyBase(y)// new threshold
return(τ; (x, Counters[x]) for x in Counters)
```

Simulations

CV 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 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 ℓ.

Simulation Results for ℓ-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 ℓ-capped sample.

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

ℓ, 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

ℓ, 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 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 ($\Longrightarrow$ 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.

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

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
  - ℓ-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.

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
  - ℓ-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.

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.

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

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.

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

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

Bibliography II

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

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 and G. N. Martin.
Probabilistic counting algorithms for data base applications.
J. Comput. System Sci., 31:182–209, 1985.

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 III

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.

E. Ohlsson.
Sequential poisson sampling.
J. Official Statistics, 14(2):149–162, 1998.

Bibliography IV

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.