Stream Sampling for Frequency Cap Statistics

Edith Cohen
Google Research, CA, USA
Tel Aviv University, Israel
edith@cohenwang.com

ABSTRACT

Unaggregated data, in a streamed or distributed form, is prevalent and comes from diverse sources such as interactions of users with web services and IP traffic. Data elements have *keys* (cookies, users, queries) and elements with different keys interleave.

Analytics on such data typically utilizes statistics expressed as a sum over keys in a specified segment of a function f applied to the frequency (the total number of occurrences) of the key. In particular, *Distinct* is the number of active keys in the segment, *Sum* is the sum of their frequencies, and both are special cases of *frequency cap* statistics, which cap the frequency by a parameter T. One important application of cap statistics is staging advertisement campaigns, where the cap parameter is the limit of the maximum number of impressions per user and we estimate the total number of qualifying impressions.

The number of distinct active keys in the data can be very large, making exact computation of queries costly. Instead, we can estimate these statistics from a sample. An optimal sample for a given function f would include a key with frequency w with probability roughly proportional to $f(w)$. But while such a "gold-standard" sample can be easily computed over the aggregated data (the set of key-frequency pairs), exact aggregation itself is costly and slow. Ideally, we would like to compute and maintain a sample without aggregation.

We present a sampling framework for unaggregated data that uses a single pass (for streams) or two passes (for distributed data) and state proportional to the desired sample size. Our design unifies classic solutions for Distinct and Sum. Specifically, our ℓ-capped samples provide nonnegative unbiased estimates of any monotone non-decreasing frequency statistics, and close to gold-standard estimates for frequency cap statistics with $T = \Theta(\ell)$. Furthermore, our design facilitates multi-objective samples, which provide tight estimates for a specified set of statistics using a single smaller sample.

1. INTRODUCTION

The data available from many services, such as interactions of users with Web services or content, search logs, and IP traffic, is presented in an *unaggregated* form. In this model, each data *element* has a *key* from a universe $\mathcal{X}$ and a weight $w > 0$. Data elements with different keys interleave in a data stream or distributed storage.

The *aggregated view* of the data is a set of pairs that consists of an *active* keys $x \in \mathcal{X}$ (a key that occurred at least once) and the respective total weight w_x of all elements with key x. When all element weights are uniform, w_x is the number of occurrences of an element with key x in the data. The weight w_x is often referred to as the *frequency* of the key (it is proportional to the actual frequency in the data set).

Frequency statistics of such data are fundamental to data analytics. Queries have the form

$$Q(f, H) \equiv \sum_{x \in \mathcal{X} \cap H} f(w_x) \,, \tag{1}$$

where $f(w) \geq 0$ is a nonnegative function such that $f(0) = 0$ and H is a selection predicate that specifies a segment of the key population $\mathcal{X}$. Typically f is monotone non decreasing, which means that more frequent keys carry at least the same contribution as less frequent ones. Some prominent examples are the p^{th} *frequency moment*, where $f(x) = x^p$ for $p > 0$ [1] and *frequency cap* statistics, where f is a *cap* function with parameter $T > 0$:

$$\mathsf{cap}_T(y) \equiv \min\{y, T\} \,.$$

Two special cases of both cap statistics and frequency moments, that are widely studied and applied in big data analytics, are *Distinct* – the number of distinct (active) keys in the segment (L_0 moment or cap_1, assuming elements weights are ≥ 1), and *Sum* – the sum of weights of elements with keys in the segment (L_1 moment or cap_∞).

Frequency caps that are in the mid-range mitigate the domination of the statistics by the (typically few) very frequent keys but still provide a larger representation of the more frequent keys. Mid-range frequency cap statistics are prevalent in online advertising platforms [17, 28]: A common practice is to allow an advertiser to specify a limit to the number of impressions of an ad campaign that any individual user is exposed to in a particular duration of time. Advertisements also typically target only a segment H of users (say certain demographics and geographic location). The statistics $Q(\mathsf{cap}_T, H)$ is the number of qualifying opportunities for placing an ad. These queries are posed over past data in order to provide an advertiser with a prediction for the total potential number of qualifying impressions. Often, the prediction needs to be computed or estimated quickly, to facilitate interactive campaign planning.

An exact computation of frequency statistics (1) requires aggregating the data by key. The representation size of the aggregated view, however, and the runtime state needed to produce it, are linear in the number of distinct keys. Often, the number of distinct keys is very large and our system can be using the same resources to process many different streams or workloads. To scalably mine such data, our computation needs to be limited to one or few passes over elements while maintaining a small runtime state (which translates to memory or communication). A single pass (stream computation) is necessary when the data is discarded (such as with IP traffic) or

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author.
KDD'15, August 10-13, 2015, Sydney, NSW, Australia.
ACM 978-1-4503-3664-2/15/08.
http://dx.doi.org/10.1145/2783258.2783279.

when statistics are collected for live dashboards. Under these constraints, we often must settle for a small summary of the data set which can provide approximate answers [14, 16, 1].

A solution which only addresses the final summary size is to first compute the aggregated view, and then retain a sample for future queries: For each key we compute a weight equal to $f(w_x)$, and we then apply a weighted sampling scheme such as Probability Proportion to Size (pps) [30], VarOpt [2, 9], or bottom-k, which includes *successive weighted sampling without replacement* (ppswor) and sequential Poisson/Priority sampling [29, 27, 12].

From the weighted sample we can compute *approximate* segment frequency statistics by applying an appropriate *estimator* to the sample. There is a well-understood tradeoff between the sample size k and the accuracy of our approximation. For a segment H that has proportion $q = Q(f, H)/Q(f, \mathcal{X})$ of the statistics value on the general key population, the coefficient of variation (CV) is (roughly) $(qk)^{-0.5}$: the inverse of the square root of qk. The CV is the standard error normalized by the mean, and corresponds to the NRMSE (normalized root mean square error). That is, in order to obtain NRMSE of $\epsilon = 0.1$ (10%) on segments that have at least $q = 0.001$ fraction of the total value of the statistics, we need to choose a sample size of $k = \epsilon^{-2}/q = 10^5$, which is usually much smaller than the number of distinct keys we might have. This also means that we can obtain confidence intervals on our estimates using the actual number of samples from our segment. Moreover, this CV bound is the best we can hope for (on average over segments) and will be the gold standard we use in the design of sampling schemes and estimators in more constrained settings, which preclude aggregating the data.

The challenge is to produce an effective sample of the data, using one or few passes while maintaining state that ideally is of the order of the desired sample size. There is a large body of work on stream sampling schemes designed for distinct and sum queries. The Sample and Hold (SH) family of sampling schemes [16, 13, 6, 8] and another based on VarOpt [7] are suited for sum queries. Distinct reservoir sampling of keys [24, 14] is suited for distinct queries. Both SH [8] and distinct sampling support unbiased estimates of all frequency statistics and meet our $(qk)^{-0.5}$ CV upper bound target for the particular statistics they are designed for (the claim for SH is established here). They do not provide, however, comparable statistical guarantees for other statistics.

Contributions and Road Map

Our main contribution is a general sampling framework for unaggregated streams. The sampling scheme is specified by a random scoring function that is applied to stream elements. The scoring function is tailored to the statistics we want to estimate. Our framework is presented in Section 3 and we cast the existing distinct and SH sampling schemes as special cases.

Our framework facilitates the first stream sampling solution with CV upper bound that is close to the $(qk)^{-0.5}$ gold standard for general frequency cap statistics. We offer two basic designs: A discrete spectrum that only handles uniform elements and a continuous spectrum which handles arbitrary positive element weights.

Our discrete spectrum is presented in Section 4. The sampling algorithms SH_ℓ are parametrized by an integer *cap* parameter ℓ. When ℓ exceeds the maximum frequency over keys, SH_ℓ is equivalent to classic SH. For $\ell = 1$, it is identical to distinct sampling. We derive unbiased and admissible estimators for any *discrete* frequency statistics, that is, f specified for nonnegative integers.

Our continuous spectrum SH_ℓ, for a positive real cap parameter ℓ, is presented in Section 5. When $\ell \gg \max_x w_x$, SH_ℓ is identical to weighted SH [6]. For $\ell \ll \min_x w_x$, SH_ℓ is distinct sampling. We derive estimators of frequency statistics where the function f is continuous and differentiable almost everywhere. Note that most natural statistics, including frequency moments and cap statistics, can be expressed as continuous monotone functions, which are differentiable almost everywhere. Surprisingly perhaps, the continuous spectrum, which may seem less intuitive than the discrete spectrum, yields an elegant and simple specification of estimators.

We show that our estimates of cap_T statistics from SH_ℓ samples have CV upper bounded by $O((qk)^{-0.5})$ when $T = O(\ell)$. The CV bound gracefully degrades with *disparity* $\max\{T/\ell, \ell/T\}$ between the sample cap parameter ℓ and the statistics cap parameter T. The estimate of any frequency function f is unbiased and for f that is monotone non-decreasing, also nonnegative. This makes our design very versatile.

Our estimators are derived by expressing sampling as a transform from frequencies to expected "sampled frequencies," and then inverting the transform. The transform is a matrix vector product In the discrete case and an integral transform in the continuous case. For the latter, the estimator is a simple expression in terms of f and its derivative f'. Since our estimators are the unique inverse of the transform, they are the minimum variance unbiased nonnegative estimators for the sampling scheme, meaning that in terms of variance, they optimally use the information in the sample. Our discrete estimators generalize a matrix inversion applied in [19, 8] to estimate the flow size distribution from Sampled Netflow and SH IP flow records. Our continuous spectrum estimators are novel even for the basic weighted SH scheme, for which previously only estimators for sum statistics were provided [6].

In Section 6 we propose a design for multi-objective sampling [11] that addresses applications that require estimates with statistical guarantees for multiple, possibly all, cap statistics.

Our proposed sampling algorithms and estimators are simple and highly practical, despite a technical analysis. The application resembles that of classic (uncapped) SH, distinct sampling, and approximate distinct counting algorithms that are prevalent in industrial applications [18]. Section 7 includes an experimental evaluation which demonstrates superior accuracy versus sample size tradeoffs by using a sample that is suited for the statistic. Due to a page limit, we could not include most proofs and many details and refer the reader to the technical report [5].

2. PRELIMINARIES

We work with key value data sets that consist of *elements* (x, w), where x is a key from a universe $\mathcal{X}$ and $w > 0$. The data set is *aggregated* if each key appears in at most one element and is *unaggregated* otherwise. We define the weight $w_x \equiv \sum_{h \in x} w(h)$ of a key x to be the sum of the weights of elements with key x. If x is not active (there are no elements with key x), we define $w_x = 0$. When element weights are uniform, we define w_x to be the number of elements with key x. The *aggregated view* of an unaggregated data set has elements (x, w_x) for all active keys x.

We are interested in sampling algorithms that process the unaggregated data in one or few passes while maintaining state that is proportional to the sample size. Such algorithms can be scalably executed when elements are streamed (presented sequentially to the algorithm) or distributed across multiple locations.

We start with a quick review of relevant sampling schemes for aggregated data sets. A Poisson sample of a key value dataset $\{(x, w_x)\}$ is specified by sampling probabilities p_x. The sample S includes each $x \in \mathcal{X}$ with independent probability p_x and has expected size $\mathsf{E}[|S|] = \sum_x p_x \equiv k$. To estimate a frequency statistics $Q(f, H)$ from the sample, we can apply the inverse probability estimator $\hat{Q}(f, H) = \sum_{x \in H \cap S} f(w_x)/p_x$ [20]. This estimator

can be interpreted as a sum of per-key estimates that are $f(w_x)/p_x$ if $x \in S$ and 0 otherwise. Note that this estimator can only be applied when w_x and p_x are available for all $x \in S$. It is nonnegative and is unbiased if $p_x > 0$ when $f(w_x) > 0$. It is actually the minimum variance unbiased and nonnegative *sum* estimator (sum of per-key estimates) for the given probabilities $\{p_x\}$.

For a dataset $\{(x, w_x)\}$, function f, and (expected) sample size k, one can ask what are the "optimal" sampling probabilities. It is well known that if we sample keys with probability proportional to their contribution $f(w_x)$ (pps), we minimize the sum of per-key variances $\sum_x f(w_x)^2(1/p_x - 1)$. With pps, we have the following statistical guarantee: For estimates of the statistics $Q(f, H)$, where the segment H has proportion

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

of the statistics, the variance of our estimate is

$$\mathsf{var}[\hat{Q}(f,H)] \leq \frac{1}{qk} Q(f,H)^2 \ .$$

Thus the CV (normalized standard error) is at most $(qk)^{-0.5}$, which is the best bound we can hope for on average over segments with proportion q. That is, any scheme that would do better on some segments, would do worse on others. Other weighted sampling schemes we mentioned in the introduction provide this statistical guarantee with a fixed sample size k.

One of these schemes that is particularly relevant for our treatment of unaggregated data sets is ppswor: Keys are drawn successively so that at each step the probability that we draw x is proportional to its weight relative to the remaining unsampled keys: $f(w_x)/\sum_{y\notin S} f(w_y)$. The sampling can be realized by associating with each key a random $\texttt{seed}(x) \sim \mathsf{Exp}[f(w_x)]$ (exponentially distributed seed with parameter $f(w_x)$) [29]. Ordering keys by increasing seed value turns out to correspond exactly to ppswor sampling order. A *fixed-threshold* sample, for a pre-specified threshold τ, includes all keys with $\texttt{seed}(x) < \tau$. Alternatively, we can obtain a *fixed size* (bottom-k) sample by taking the k keys with smallest seed values. In the latter case, it is convenient to define τ as the $(k+1)$ smallest seed.

Finally, we can estimate a statistics $Q(g, H)$ from the ppswor sample taken for weights $f(w_x)$ as follows. When we use fixed threshold sampling, we compute the probability that x is sampled

$$\Phi_\tau(w_x) \equiv \Pr[\texttt{seed}(x) < \tau] = 1 - e^{-f(w_x)\tau} \ ,$$

and apply inverse probability:

$$\hat{Q}(g,H) = \sum_{x\in H\cap S} \hat{g}(w_x \mid \tau), \text{ where } \hat{g}(w_x \mid \tau) \equiv \frac{g(w_x)}{\Phi_\tau(w_x)} \ . \tag{2}$$

Note that $\Phi_\tau(w_x)$ only depends on w_x and τ (which are available for sampled keys). When we work with a fixed sample size k and define τ to be the $(k+1)$ smallest seed, we can interpret $\Phi_\tau(w_x)$ as the probability that the key x is sampled, conditioned on fixed randomization of other keys. This means that the estimator (2) is unbiased [10]. Moreover, the estimates $\hat{g}(w_x \mid \tau)$ obtained for different keys x have zero covariances [10], which allows us to bound the variance on segment queries as we would do when sampling with a pre-specified threshold. It turns out (see TR[5]) that for statistics $\hat{Q}(f, H)$ with proportion q, the CV is at most $(q(k-1))^{-0.5}$, which is essentially (within a single sample) our "gold standard" CV.

A ppswor sample with respect to any function $f(w_x)$ can be computed from a streamed (or distributed) aggregated data $\{(x, w_x)\}$, using state proportional to the sample size. This is not generally possible, however, over unaggregated data: For example, there are polynomial lower bounds on the state needed by a streaming algorithm which approximates frequency moments $Q(x^p, \mathcal{X})$ with $p > 2$ [1].

3. SAMPLING FRAMEWORK

We present a framework for sampling unaggregated data sets and cast SH and distinct sampling in our framework. Our algorithms compute a sample S while maintaining state, in the form of a cache S of sampled keys, that is proportional to the sample size. Each sampling scheme in specified through a random mapping $\texttt{ElementScore}(h)$ of elements $h = (x, w)$ to numeric score values. The distribution of $\texttt{ElementScore}$ may only depend on the key x and w. We then define the *seed* of a key x

$$\texttt{seed}(x) = \min_{h \text{ with key } x} \texttt{ElementScore}(h) \tag{3}$$

to be the random variable that is the minimum score of all its elements. As with ppswor, we can obtain a fixed-threshold sample $S = \{x \mid seed(x) < \tau\}$, which for a given τ includes all keys with $\texttt{seed}(x) < \tau$, or a fixed-size sample, which for a specified sample size k includes the k keys with smallest seed values and define τ to be the $(k+1)$ smallest seed.

Once we have the sample, we can apply estimators to it to approximate statistics. To do so, we need to have information on the weight of sampled keys. The exact weights w_x of sampled keys $x \in S$ can be computed in a second pass over the data, as we detail below. We also consider a pure streaming (single sequential pass) setting, where we generally settle for some $c_x \leq w_x$ and we derive estimators that are able to work with this information. In terms of computation platform, our 2-pass schemes can be fully parallelized or distributed whereas our 1-pass (streaming) schemes are not as flexible: They can be executed on multiple streams that are processed separately (as with sharding) provided that all elements with the same key are processed at the same shard.

3.1 2-pass scheme

The first pass identifies the set of keys S with smallest seeds. For fixed-threshold sampling, our summary contains all keys with scores below τ. With fixed-size sampling, the summary contain the keys with k smallest minimum scores. These summaries are mergeable, that is, from the summaries of two data sets we can compute a summary of their union. For fixed-τ, the merged summary is the union of the keys in the two summaries. For fixed-k, we compute the seed of each key in the union as the minimum seed attained in each of the summaries. We then take the k keys with smallest seeds (retaining their seed values) as a summary of the union. Either way the summary sizes never exceed $|S|$, which is the final sample size. The second pass, which computes w_x for $x \in S$ uses summaries that are the weight of each key $x \in S$ the data set. We merge two summaries by key-wise addition of weights to obtain a summary of the union. Algorithm 1 is 2-pass stream sampling of a fixed sample size k.

3.2 Fixed threshold stream sampling

A fixed threshold scheme processes an element $h = (x, w)$ as follows. If $x \in S$ (key x is cached/sampled), then $c_x \leftarrow c_x + w$. Otherwise, if $\texttt{ElementScore}(h) < \tau$, then x is inserted to S and $c_x \leq w$ is initialized. The discrete scheme which applies to uniform weights $w = 1$, is provided as Algorithm 2, and uses the initialization $c_x \leftarrow 1$ ($\mathsf{Counters}[x]$ in the pseudocode). A continuous scheme is presented in Section 5.

Algorithm 1: 2-pass stream sampling: fixed size k

Data: sample size k, elements (x, w) where $x \in \mathcal{X}$ and $w > 0$
Output: set of k pairs (x, w_x) where $x \in \mathcal{X}$

```
Counters ← ∅ // Initialize sample
τ ← +∞ // Upper bound on ElementScore
// Pass I: Identify the k sampled keys
foreach stream element h = (x, w) do
    if x is in Counters then
        seed(x) ← min{seed(x), ElementScore(h)}
    else
        s ← ElementScore(h)
        if s < τ then
            seed(x) ← s; Counters[x] ← 0
            if |Counters| = k + 1 then
                y ← arg max{seed(x) | x in Counters}
                τ ← seed(y)
                delete seed(y), Counters[y]
// Pass II: Compute w_x for sampled keys
foreach stream element h = (x, w) do
    if x is in Counters then
        Counters[x] ← Counters[x] + w
return (τ ; (x, Counters[x]) for x in Counters)
```

Algorithm 2: Stream sampling with fixed threshold τ

Data: threshold τ, stream of elements with key $x \in \mathcal{X}$
Output: set of pairs (x, c_x) where $x \in \mathcal{X}$ and $c_x \in [1, w_x]$

```
Counters ← ∅ // Initialize Counters cache
foreach stream element h with key x do // Process a stream element
    if x is in Counters then
        Counters[x] ← Counters[x] + 1;
    else
        if ElementScore(h) < τ then
            Counters[x] ← 1; // Initialize c_x
return ((x, Counters[x]) for x in Counters)
```

3.3 Fixed size stream sampling

Algorithm 3 provides pseudocode for discrete (uniform weights) stream sampling with a fixed sample size k.

The algorithm maintain a set S (Counters) of cached keys. For each cached key x, it keeps a count c_x (Counters[x]) and a lazily computed seed value $\texttt{seed}(x)$. When processing an element h with key x, we compute $y \leftarrow \texttt{ElementScore}(h)$. If $x \in S$, we increment c_x.

Otherwise, if $x \notin S$ and $y < \tau$, we insert $x \in S$ with $c_x \leftarrow 1$ and $\texttt{seed}(x) \leftarrow y$. As a result, we may have $|S| = k + 1$ cached keys. In this case, we would like to evict from S the key with maximum seed. But the seeds are not fully evaluated yet, in that the current $\texttt{seed}(x)$ only reflect the seed up to the first element that is currently counted in c_x.

We repeat the following until a key is evicted. We pop from S the key y with maximum current seed and set $\tau \leftarrow \texttt{seed}(y)$. We then iterate decreasing the count c_y and scoring "uncounted" elements until either the count becomes $c_y = 0$ and y is evicted or we obtain a score that is below τ. In the latter case, we reinsert y to S with $\texttt{seed}(y)$ equal to that score.

Algorithm 3: Stream sampling with fixed size k

Data: sample size k, stream of elements with key $x \in \mathcal{X}$
Output: set of k pairs (x, c_x) where $x \in \mathcal{X}$ and $c_x \in [1, w_x]$

```
Counters ← ∅ // Initialization
τ ← 1 // Supremum of ElementScore range
foreach element h with key x do
    if x is in Counters then
        Counters[x] ← Counters[x] + 1
    else
        score ← ElementScore(h)
        if score < τ then
            seed(x) ← score
            Counters[x] ← 1
            while |Counters| > k do
                y ← arg max{seed(x) | x in Counters}
                τ ← seed(y)
                while Counters[y] > 0 and seed(y) ≥ τ
                do
                    Counters[y] ← Counters[y] − 1
                    seed(y) ← ElementScore(y)
                if Counters[y] == 0 then
                    delete Counters[y], seed(y)
return (τ ; (x, Counters[x]) for x in Counters)
```

Analysis. Clearly, the work of Algorithm 2 (fixed-threshold sampling) is $O(1)$ per stream element. We show amortized work $O(1)$ for Algorithm 3 (fixed-size sampling). (See TR [5] for proof.)

3.4 Element scoring properties

We will select the element scoring function according to the statistics f we are interested in. Intuitively, to obtain quality estimates (CV upper bound of $(qk)^{-0.5}$), we would need to sample each key x with probability roughly proportional to $f(w_x)$. The challenge is to identify when and how we can achieve this by a small state streaming algorithm.

Some properties of our element scoring functions that greatly simplify the derivation of estimators are that seed values of different keys are independent and that for a particular key x, the distribution of $\texttt{seed}(x)$ (the minimum element score) depends only on w_x, and not on the arrangement of elements or on the breakdown of the weight of each key to different elements. Furthermore, we would also want the distribution of c_x for $x \in S$ to only depend on w_x and τ. We assume here that we work with perfectly random numbers and hash functions.

3.5 Estimation

As with the ppswor estimator reviewed in Section 2, we use estimators that can be expressed as a sum over keys $x \in H$ of individual estimates $\hat{f}(w_x)$ of $f(w_x)$. The estimate are unbiased and are 0 for keys $x \notin S$.

With two-pass sampling (Algorithm 1), we have the weight w_x, and therefore $f(w_x)$, for each sampled key $x \in S$. When the seed distribution only depends on w_x and τ, we can compute the inclusion probability $\Phi_\tau(w_x)$ of a key x from its weight w_x and apply inverse probability estimation as in (2). In the streaming (single pass) schemes, the sample includes a partial count $c_x \leq w_x$ for each $x \in S$. The requirement that the distribution of c_x only depends on w_x and τ allows us to express sampling as a transform

(which depends on τ) from the distribution w_x to the expected outcome distribution c_x. The derivation of unbiased estimators then corresponds to inverting this transform.

The transforms we obtain have a unique inverse, which means that our estimators are the optimal (minimum variance) unbiased and nonnegative sum estimators. Because the 2-pass estimators (2) are also optimal, and rely on more information – the exact value w_x instead of a sample from a distribution with parameter w_x, the variance of the streaming estimators is always at least that of the 2-pass estimator.

The estimators for both the fixed-threshold and the fixed sample-size schemes are stated in terms of the threshold probability τ. When working with a fixed sample-size k, τ is defined as the $(k+1)$st smallest seed. As with ppswor, τ when defined this way plays the same role as the threshold value τ used in a fixed sampling threshold scheme: The probability that a key is sampled, conditioned on fixed randomization of other keys, is the probability that its seed value is below the kth smallest seed of other keys. When the key is included in the sample, this value is τ. Similarly, under the same conditioning, the distribution of c_x only depends on τ and w_x, and is the same one as the respective fixed threshold scheme with τ. Moreover, the covariance of the estimates obtained for two different keys x, y is zero. The argument is the same as with ppswor [10] and SH [8]. This important property allows us to bound the variance on estimates of segment statistics by the sum of variance of estimates for individual keys.

We now cast two existing basic sampling schemes in our framework: Distinct, which is designed for cap_1 statistics and SH, which is designed for cap_∞ (sum) statistics.

3.6 Distinct sampling

A distinct sample is a uniform sample of active keys (those with $w_x > 0$), meaning that conditioned on sample size k, all subsets of active keys are equally likely. For an element h with key x, we use $\texttt{ElementScore}(h) = \texttt{Hash}(x)$, where $\texttt{Hash}(x) \sim U[0,1]$ is a random hash function selected before we process the stream. Note that all elements of the same key x have the same score and therefore $\texttt{seed}(x) \equiv \texttt{Hash}(x)$.

When we sample with respect to a fixed threshold τ, we retain all keys with $\texttt{Hash}(x) < \tau$. When using a fixed sample size k, the scheme is the following (distinct variant) of reservoir sampling [24]: For each stream element, compute $\texttt{Hash}(x)$ and retain the k keys with smallest hash values.

With distinct sampling, the value c_x is equal to the exact weight w_x for each sampled key x. This is because any key that enters our cache does so on the first element of the key. If a key is evicted, (in the fixed k scheme), it can never re-enter. We also have that for all keys with $w_x > 0$, the probability that x is sampled is $\Phi_\tau(w_x) \equiv \tau^{-1}$. We can therefore apply the inverse probability estimator (2):

$$\hat{Q}(f,H) = \tau^{-1} \sum_{x \in S \cap H} f(w_x)\,. \tag{4}$$

Distinct sampling is optimized for distinct (cap_1) statistics. In particular, $\hat{Q}(\mathsf{cap}_1, \mathcal{X})$ has CV upper bounded by $(k-1)^{-0.5}$ [3, 4] and for a segment H with proportion q of distinct keys, $\hat{Q}(\mathsf{cap}_1, H)$ has CV upper bounded by $(q(k-1))^{-0.5}$, as it is equivalent to the ppswor estimator for $f(w) \equiv 1$. For general cap_T statistics, however, the CV grows rapidly with T (we shall see it is $\propto \sqrt{T}$). This is because our uniform sample of active keys can easily miss keys with high $f(w_x)$ values which contribute more to the statistics.

3.7 Sample and Hold (SH)

Classic SH, with fixed sampling threshold τ or with fixed sample size k, [16, 13] is specified for uniform element weights, so that w_x is the number of elements with key x. We cast SH in our framework using $\texttt{ElementScore}(h) \sim U[0,1]$. Note that each key x can have many independent scores drawn, one for each element of x. Therefore, the more elements a key has, the more likely it is to be sampled. The seed is the minimum element score, which can be transformed to an exponentially distributed random variable with parameter w_x. Therefore, as observed in [8], the SH sample is actually a ppswor sample with respect to the weights w_x [29]. When we use a second pass (Section 3.1) to obtain the exact weights w_x, we can apply the ppswor estimator (2).

With stream sampling (Algorithm 2 and Algorithm 3), the final count of a key x has $c_x \leq w_x$, where $w_x - c_x + 1$ is geometric with parameter τ, truncated at $w_x + 1$ (probability of $c_x = 0$ is $(1-\tau)^{w_x}$). An unbiased estimator for statistics $Q(f,H)$ from an SH sample is [8]:

$$\hat{Q}(f,H) = \tau^{-1} \sum_{x \in S \cap H} \Big(f(c_x) - f(c_x - 1)(1-\tau) \Big)\,. \tag{5}$$

Note that this estimator is nonnegative when f is monotone non decreasing. This is because for all $i > 0$, $f(i) - f(i-1)(1-\tau) > 0$. Surprisingly perhaps, we show here (see TR [5]) that the 1-pass estimate is not too far from the 2-pass estimate in that for sum statistics ($f(x) = x$) the CV is also upper bounded by $(q(k-1))^{-0.5}$.

For cap statistics with small T, however, the SH estimates can have CV that far exceeds our $(qk)^{-0.5}$ target: When the frequency distribution is highly skewed, the ppswor sample would be dominated by heavy keys. This means that segments with a large proportion of the cap_T statistics that mostly include keys with low frequencies would have a disproportionally small representation in the sample and thus large errors.

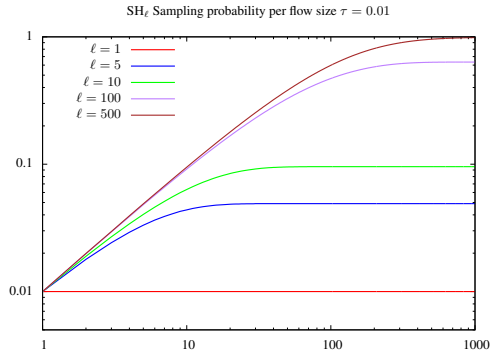

Figure 1: SH$_\ell$ sampling probability per key weight w, for selected values of ℓ ($\tau = 0.01$). Note that for $w \gg \ell \log \ell$ probability is constant and for $w \ll \ell$, probability is proportional to w. We can see that the probability is close to being proportional to $\min\{w, \ell\}$, which is what we want for estimating cap_ℓ statistics.

4. THE DISCRETE SH SPECTRUM

Our discrete SH spectrum is parametrized by an integer $\ell \geq 1$. Distinct sampling is SH$_1$ and classic SH is SH$_\infty$. In general, SH$_\ell$ is designed to estimate well frequency cap statistics with $T \approx \ell$.

The SH_ℓ element scoring function for an element h with key x draws a uniform random bucket $b \sim U[1, \cdots, \ell]$ and returns a hash of the pair $\mathtt{Hash}(x, b) \sim U[0, 1]$. Note that the buckets are independent for different elements with key x.

$$\mathtt{ElementScore}(h) \leftarrow \mathtt{Hash}(\lfloor(\ell * \mathtt{rand}())\rfloor, x)\,. \quad (6)$$

Recall that $\mathtt{seed}(x)$ (3) is the minimum score of an element with key x. When $\ell = 1$, the seed distribution is uniform for all keys with $w_x > 0$. More generally, we can see that the element scoring (6) provides up to ℓ "independent" attempts for each key to obtain a lower seed. That way, keys with more elements are more likely to have a lower seed and be sampled, but with diminishing return: Keys where $w_x \ll \min\{\ell, \tau^{-1}\}$ are sampled with probability roughly proportional to w_x whereas keys with $w_x \gg \min\{\ell, \tau^{-1}\}$ have a roughly constant inclusion probability regardless of frequency. Also note that when the cap parameter is large relative to the inverse sampling threshold $\ell \gg \tau^{-1}$, SH_ℓ is similar to SH_∞. Figure 1 illustrates these properties by showing the sampling probability of a key as a function of w_x, for selected values of the parameter ℓ.

4.1 Estimators for discrete SH_ℓ

The output of our stream sampling algorithm is a threshold value τ and a set S of pairs of the form (y, c_y), where $y \in \mathcal{X}$ and $c_y \in [1, w_y]$.

Coefficient form. We express our estimators as vectors $\boldsymbol{\beta}^{(f,\tau,\ell)}$, which depends on f, the threshold τ, and the parameter ℓ. The cth entry $\boldsymbol{\beta}_c^{(f,\tau,\ell)}$ is the contribution to the estimate of a key with count c. The estimate on the statistics $Q(f, H)$ is then

$$\hat{Q}(f, H) = \sum_{x \in S \cap H} \beta_{c_x}\,. \quad (7)$$

The distinct sample ($\ell = 1$) estimator (4) is expressed using $\beta_i \equiv f_i \tau^{-1}$ (using the notation $f_i \equiv f(i)$) whereas the SH estimator ($\ell = +\infty$) (5) is expressed using $\beta_i \equiv \tau^{-1}\Big(f_i - f_{i-1}(1-\tau)\Big)$. We seek estimators of this form for general ℓ that are unbiased, admissible, and nonnegative $\boldsymbol{\beta} \geq 0$ when f is non-decreasing.

Probability vector $\boldsymbol{\phi}$. Let ϕ_i be the probability that the ith element of the same key was the first one to get counted by SH_ℓ. The vector $\boldsymbol{\phi}$ depends on the parameters ℓ and τ.

For $\ell = 1$, we have the closed form $\phi_1 \equiv \tau$ and $\phi_i = 0$ for $i > 1$. For $\ell = +\infty$, we have $\phi_i = (1-\tau)^{i-1}\tau$.

To express $\boldsymbol{\phi}$ for general ℓ, we let a_{ij} be the probability that we used exactly $j \leq \min\{\ell, i\}$ buckets in the first i elements of a key.

By definition $a_{0i} \equiv 0$ when $i \geq 1$, $a_{ij} \equiv 0$ when $j > \min\{\ell, i\}$, and $a_{1,0} = 0$. Otherwise, $a_{1,1} = 1$ and for $i > 1$, $j \leq \min\{\ell, i\}$, the values can be computed from the relation

$$a_{ij} = a_{i-1,j}\frac{j}{\ell} + a_{i-1,j-1}\frac{\ell - j + 1}{\ell}\,. \quad (8)$$

We can now write

$$\phi_i = \tau \sum_{j=1}^{\min\{i-1,\ell-1\}} a_{i-1,j}(1-\tau)^j \frac{\ell - j}{\ell}\,.$$

A 2-pass estimator. The probability that a key x is sampled (illustrated in Figure 1) is

$$\Phi_{\tau,\ell}(w_x) \equiv \sum_{j=1}^{w_x} \phi_j\,.$$

If we use a 2-pass scheme (Section 3.1), we can apply the inverse probability estimator (2) $\hat{Q}(f, H) = \sum_{x \in S \cap H} \frac{f(w_x)}{\Phi(w_x)}$.

Inverting the sample counts. We now derive a streaming estimator. We use the notation $o_i = \{x \in S \cap H \mid c_x = i\}$ (the "observed" count) for the random variable that is the number of keys $x \in S \cap H$ with $c_x = i$. Let $m_i = \{x \in H \mid w_x = i\}$ be the number of keys in H with count $w_x = i$. Our statistics (1) can be expressed as $Q(f, H) = \boldsymbol{f}^T \boldsymbol{m}$. We have the relation $\mathsf{E}[o_i] = \sum_{j \geq i} \phi_{j-i+1} m_i$ and can write

$$\mathsf{E}[\boldsymbol{o}] = Y^{(\boldsymbol{\phi})}\boldsymbol{m}\,.$$

We use the notation $Y^{(\boldsymbol{v})}$ for an upper triangular matrix that corresponds to a vector $\boldsymbol{v}$, such that $\forall, j \geq i$, $[Y^{(\boldsymbol{v})}]_{ij} \equiv v_{j-i+1}$.

We have $\boldsymbol{m} = (Y^{(\boldsymbol{\phi})})^{-1}\mathsf{E}[\boldsymbol{o}]$. Therefore, from linearity, $\hat{\boldsymbol{m}} \equiv (Y^{(\boldsymbol{\phi})})^{-1}\boldsymbol{o}$ is an unbiased estimator of $\boldsymbol{m}$. Therefore, to compute the estimate we need to invert $Y^{(\boldsymbol{\phi})}$.

The inverse of the matrix $Y^{(\boldsymbol{\phi})}$ has the same upper triangular structure, and can be expressed as $Y^{(\boldsymbol{\psi})}$ with respect to another vector $\boldsymbol{\psi}$. To compute $\boldsymbol{\psi}$, we consider the constraints $Y^{(\boldsymbol{\psi})}Y^{(\boldsymbol{\phi})} = I$ obtained from the product of the first row of $Y^{(\boldsymbol{\psi})}$ with the columns of $Y^{(\boldsymbol{\phi})}$. We obtain the equations $\psi_1 = \phi_1^{-1}$, and for $j > 1$,

$$\sum_{j=1}^{i} \psi_j \phi_{1+i-j} = 0\,.$$

This allows us to iteratively solve for ψ_i after computing ψ_j for $j < i$ using

$$\psi_i = \phi_1^{-1}(-\sum_{j=1}^{i-1} \phi_{1+i-j}\psi_j)\,.$$

For distinct sampling we have $\psi_1 = \tau^{-1}$ and $\psi_i = 0$ for $i > 1$. For SH [8] we have $\psi_1 = \tau^{-1}$, $\psi_2 = -(1-\tau)\tau^{-1}$, and $\psi_i = 0$ for $i \geq 2$. In general, however, $\boldsymbol{\psi}$ can have many non-zero entries.

We show the following (see TR [5]):

THEOREM 4.1. *The estimator $\hat{Q}(f, H) = \sum_{x \in S \cap H} \beta_{c_x}$, where*

$$\beta_i^{(f,\tau,\ell)} \equiv \sum_{j=1}^{i} \psi_j f_{i-j+1}$$

is unbiased.

We show that the estimates are nonnegative when f is monotone non-decreasing:

THEOREM 4.2. *When f is monotone non-decreasing, then for all ℓ and τ, $\boldsymbol{\beta}^{(f,\tau,\ell)} \geq 0$.*

5. THE CONTINUOUS SH SPECTRUM

We now present our continuous SH_ℓ sampling schemes, which generalizes SH with weighted updates ($\ell = \infty$) [6]. The continuous design offers the following advantages over the discrete design even when applied to uniform weights. First, fixed sample-size sampling no longer requires explicitly maintain a lazy $\mathtt{seed}(x)$ for cached keys as we did in Algorithm 3: The lazy value is implicitly captured by the current threshold τ. Second, the estimator can be expressed in terms of f and its derivative. Lastly, the continuous spectrum facilitates multi-objective samples (Section 6).

Our input is a stream of elements $h = (x, w)$ with key x and a weight $w > 0$. Our element scoring is as follows: Each key has a *base hash* $\mathtt{KeyBase}(x) \sim U[0, 1/\ell]$, that is fixed for the computation and is uniformly distributed in $[0, 1/\ell]$: $\mathtt{KeyBase}(x) \leftarrow$

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

$$\texttt{ElementScore}(h) = (v \sim \mathsf{Exp}[w]) \leq 1/\ell \ ? \ \texttt{KeyBase}(x) \ : \ v \ . \tag{9}$$

The random variables $\mathsf{Exp}[w]$ are independent for different elements and are also independent of $\texttt{KeyBase}(x)$.

We now consider the distribution of $\texttt{seed}(x)$ (the minimum element score of stream elements with key x). We use properties of the exponential distribution (see TR [5] for details) and show that

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

Note that the element scoring satisfies our requirement (Section 3.4) that the distribution of $\texttt{seed}(x)$ depends only on w_x.

Qualitatively, we can see that when $w_x \ll \ell$, the seed is close to exponentially distributed with parameter w_x, and we are close to ppswor. When $w_x \gg \ell$, the seed is uniform. Intuitively, the sampling probability of keys would be roughly proportional to $\mathsf{cap}_\ell(w_x)$ which is what we need to approach the "gold standard" CV.

5.1 2-pass estimator

Consider 2-pass sampling (Section 3.1) with our element scoring function (9). For estimation, we need to compute the probability $\Phi_{\tau,\ell}(w_x) = \Pr[\texttt{seed}(x) < \tau]$ of a key with weight w_x in a sample with parameters ℓ and τ. If $\tau\ell < 1$, then a key is included if $\mathsf{Exp}[w_x] < 1/\ell$ and then $\texttt{KeyBase}(x) < \tau$. These two events are independent and have joint probability $(1 - e^{-w_x/\ell})\tau\ell$. If $\tau\ell \geq 1$ then a key is included if $\mathsf{Exp}[w_x] < \tau$, which has probability $(1 - e^{-\tau w_x})$. We can express the combined probability as

$$\Phi_{\tau,\ell}(w_x) \equiv (1 - e^{-w_x \max\{1/\ell,\tau\}}) \min\{1, \tau\ell\} \ . \tag{10}$$

We can then apply inverse probability (2) to estimate a segment statistics $\hat{Q}(f, H) = \sum_{x \in S \cap H} f(w_x)/\Phi_{\tau,\ell}(w_x)$.

We show the following (Proof provided in TR [5]):

THEOREM 5.1. *The CV of estimating $Q(\mathsf{cap}_T, H)$ from an SH$_\ell$ sample with exact weights w_x is at most*

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

Note that when $\ell = \Theta(T)$, the CV is $O((q(k-1))^{-0.5})$ and the upper bound degrades smoothly with the *disparity* $\max\{T/\ell, \ell/T\}$ between ℓ and T. Also note that the increased CV due to the constant $e/(e-1)$ and the disparity arise from a worst-case and somewhat forgiving analysis and are not inherent.

5.2 1-pass algorithms

The streaming (1-pass) algorithms compute a sample S of cached keys and a value $c_x \leq w_x$ for each $x \in S$.

Algorithm 4 performs fixed threshold sampling. When processing an element $h = (x, w)$ with a cached key, we update $c_x \leftarrow c_x + w$. Otherwise, we compute the weight Δ that would be needed for the score to be below $\max\{1/\ell, \tau\}$. If $w \leq \Delta$, we break. Otherwise, if $\tau < 1/\ell$, we break if $\texttt{KeyBase}(x) \geq \tau$. Finally, we initialize a couter $c_x \leftarrow w - \Delta$. Intuitively, the score is continuously assigned to the mass w_x. The value $c_x \leq w_x$ is the weight observed after the point in which the score gets below τ.

Fixed sample size sampling is provided as Algorithm 5. To maintain a fixed size sample, the threshold is decreased when there are $k+1$ cached keys, to the point needed to evict a key. The algorithm "simulates" the end result of working with the lower threshold to begin with.

Algorithm 4: Continuous SH$_\ell$ stream sampling: fixed τ

Data: threshold τ, stream of elements (x, w) where $x \in \mathcal{X}$ and $w > 0$
Output: set of pairs (x, c_x) where $x \in \mathcal{X}$ and $c_x \in (0, w_x]$

```
Counters ← ∅ // Initialize Counters cache
foreach stream element (x, w) do // Process a stream element
    if x is in Counters then
        Counters[x] ← Counters[x] + w;
    else
        Δ ← −ln(1−rand())/max{τ,1/ℓ} // ∼ Exp[max{τ, 1/ℓ}]
        if KeyBase(x) < min{τ, 1/ℓ} and Δ < w then
        // initialize counter for x
            Counters[x] ← w − Δ;
return ((x, Counters[x]) for x in Counters)
```

The eviction step is as follows. We draw and fix some "randomization" and compute for each cached key the threshold needed to evict the key. The randomization for key x, in the form of u_x and r_x. We then compute z_x which is the maximum threshold value that is needed to evict x with respect to that randomization. We then take the new threshold to be the maximum z_x over keys. One key (the one with maximum z_x) is evicted. For remaining keys, c_x ($\mathsf{Counters}[x]$) is updated according to the same u_x, r_x.

We elaborate on how z_x is determined when the current threshold is τ. The key x can be viewed as having a score (computed to the point the key entered the cache) that is at most τ. We can consider the distribution of the score given that it is at most τ: With the randomization, we can take it as $u_x\tau$. A necessary requirement for x to be evicted is that the new threshold τ^* is below $u_x\tau$, so we have $z_x < u_x\tau$. Conditioned on $\tau^* < u_x\tau$, we can treat this as processing an element with a new (uncached) key x and weight c_x. We consider the threshold value τ^* needed for the key to enter the cache. We simply reverse the entry rule: If $-\ln(1-r_x)/c_x \geq \ell^{-1}$, then the key would enter the cache when $\tau^* \geq -\ln(1-r_x)/c_x$. If $-\ln(1-r_x)/c_x < \ell^{-1}$, then the key would enter the cache if and only if $\tau^* \geq \texttt{KeyBase}(x)$, with count $c_x - \ell(-\ln(1-r_x))$.

We now express the distribution of c_x ($\mathsf{Counters}[x]$) and verify that it satisfies our requirement that for any key x, it only depends on w_x, ℓ, and τ. Recall that for fixed-threshold SH$_\ell$ we use the specified τ whereas with fixed-cache size SH$_\ell$, the statement is conditioned on the randomization on all other keys, which determines τ when $x \in S$. The proof is provided in TR [5].

THEOREM 5.2. *With fixed-τ SH$_\ell$ (Algorithm 4) and fixed-k SH$_\ell$ (Algorithm 5), for any key x, $c_x \sim \max\{0, w_x - \phi\}$, where ϕ has density*

$$\phi(y) = \tau \exp(-y \max\{1/\ell, \tau\})$$

in the interval $y \in [0, w_x]$.

5.3 Estimators for Continuous SH$_\ell$

We seek an unbiased and nonnegative estimator in a coefficient form, that is, a function $\beta^{(f,\tau,\ell)}(c)$ defined for any $c > 0$ and we use the estimator

$$\hat{Q}(f, H) = \sum_{x \in H \cap S} \beta^{(f,\tau,\ell)}(c_x) \ . \tag{11}$$

THEOREM 5.3. *For any continuous f that is differentiable almost everywhere, the estimator that uses*

$$\beta^{(f,\tau,\ell)}(c) \equiv f(c)/\min\{1, \ell\tau\} + f'(c)/\tau \tag{12}$$

Algorithm 5: Continuous SH_ℓ stream sampling: fixed k

Data: sample size k, stream of elements of the form (x, w) with key $x \in \mathcal{X}$ and $w > 0$
Output: τ; set of pairs (x, c_x) where $x \in \mathcal{X}$ and $c_x \in (0, w_x]$

```
Counters ← ∅; τ ← ∞ // Initialize cache
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]} // eviction
                        threshold of x
                        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] ←
                            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
                    τ ← KeyBase(y) // new
                    threshold
return (τ; (x, Counters[x]) for x in Counters)
```

is unbiased.

PROOF. We separately treat the cases where $\tau\ell < 1$ and $\tau\ell > 1$. We first show that when $\tau\ell > 1$, $\beta(c) = f(c) + f'(c)/\tau$ are unbiased estimation coefficients.

For a key of size w, we have density $\tau e^{-\tau x}$ to have count of $w - x \in (0, w)$ (otherwise the key has count 0 and the estimate is 0). We can write

$$\beta(y) = (f(y)e^{\tau y})' e^{-\tau y}\tau^{-1} .$$

Consider a key of size w. Its expected contribution to the estimate is

$$\begin{aligned}
&\int_0^w \tau e^{-\tau x}\beta(w-x)dx \\
&= \int_0^w \tau e^{-\tau x}(f(w-x)e^{-\tau(w-x)})' e^{-\tau(w-x)}\tau^{-1}dx \\
&= e^{-\tau w}\int_0^w (f(w-x)e^{-\tau(w-x)})'dx \\
&= e^{-\tau w}f(w)e^{-\tau w} = f(w)
\end{aligned}$$

We now consider the case where $\tau < 1/\ell$, showing that

$$\beta(c) = f(c)/(\ell\tau) + f'(c)/\tau$$

are unbiased estimation coefficients. For a key with weight w, we have density $\tau e^{-x/\ell}$ to have count of $w - x \in (0, w)$. We write

$$\beta(y) = (f(y)e^{y/\ell})' e^{-y/\ell}\tau^{-1} .$$

$$\begin{aligned}
&\int_0^w \tau e^{-x/\ell}\beta(w-x)dx \\
&= \int_0^w \tau e^{-x/\ell}(f(w-x)e^{(w-x)/\ell})' e^{-(w-x)/\ell}\tau^{-1} \\
&= e^{-w/\ell}\int_0^w (f(w-x)e^{(w-x)/\ell})'dx = f(w) .
\end{aligned}$$

□

Note that any continuous monotone function, including the cap_T functions, is differentiable almost everywhere and hence satisfies the requirements of the theorem. We upper bound the CV of the streaming fixed-k SH_ℓ estimator (Proof is in TR [5]):

THEOREM 5.4. *The CV of estimating $Q(\mathsf{cap}_T, H)$ from an SH_ℓ sample is upper bounded by*

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

In particular, when $\ell = \Theta(T)$, the CV is $O(q(k-1)^{-0.5})$.

6. MULTI-OBJECTIVE SAMPLES

We established that from a fixed-k SH_ℓ sample we can estimate well cap_T statistics when $T = \Theta(\ell)$. This means that if we are interested in estimates with statistical guarantees for cap values $T = [a, b]$, it suffices to use SH_ℓ samples with parameters $\ell_i = 2^i a$ for $i \leq \lceil \log(b/a) \rceil$. To process a query for a cap_T statistics, we can use the SH_ℓ sample with ℓ that is closest (within a factor of $\sqrt{2}$) from T. In particular, to estimate all cap statistics, it suffices to use $\lceil \log(\max_x w_x / \min_x w_x)) \rceil$ samples.

We now improve over this basic approach (see TR [5] for details) by instead of working with a set $\{S_\ell\}$ of samples with respective caps $\ell \in L$, we work with a single sample $S_L = \bigcup_{\ell \in L} S_\ell$. The improvement has several components: Sample coordination, which ensures that samples with closer ℓ are more similar so that $|S_L| \ll k|L|$, using estimators that benefit from the combined sample, and sampling algorithms that use state that is proportional to $|S_L|$.

7. SIMULATIONS

Our experimental evaluation is aimed to understand the error distribution of our estimators. Our analysis provided statistical guarantees on the errors that are close to the "gold standard" attainable on aggregated data. The analysis, however is worst-case in terms of the dependence on the disparity $\max\{\ell/T, T/\ell\}$, the factors of $e/(e-1) \approx 1.6$ for 2-pass and $(2e/(e-1))^{0.5} \approx 1.8$ for 1-pass, which assume a worst-case frequency distribution (error is larger when $w_x \approx \ell$), and not reflecting the advantage of with-replacement sampling that is significant when there is skew. We therefore expect actual errors to be much lower than our upper bounds.

Our sampling algorithms and estimators were implemented in Python using numpy.random and hashlib libraries. Simulations were performed on MacBook Air and Mac mini computers. We did

discrete $k = 100, \alpha = 1.2, m = 10^5, rep = 200$, NRMSE 1-pass

ℓ, T	1	5	20	50	100	500	1000	10000
1	0.098	0.115	0.185	0.279	0.374	0.658	0.862	3.016
5	**0.094**	**0.093**	0.112	0.144	0.184	0.332	0.449	1.316
20	0.133	0.111	0.102	0.109	0.122	0.199	0.254	0.615
50	0.138	0.108	**0.098**	0.101	0.107	0.163	0.207	0.419
100	0.146	0.125	0.104	**0.099**	**0.099**	0.111	0.133	0.311
500	0.171	0.135	0.123	0.112	0.110	0.102	0.101	0.149
1000	0.174	0.156	0.141	0.125	0.118	0.100	0.094	0.090
10000	0.178	0.148	0.128	0.110	0.102	**0.083**	**0.076**	**0.056**

discrete $k = 100, \alpha = 1.2, m = 10^5, rep = 200$, NRMSE 2-pass

ℓ, T	1	5	20	50	100	500	1000	10000
1	0.098	0.115	0.185	0.279	0.374	0.658	0.862	3.016
5	**0.094**	**0.093**	0.110	0.143	0.183	0.333	0.449	1.316
20	0.123	0.109	0.101	0.108	0.120	0.199	0.254	0.614
50	0.131	0.108	**0.100**	**0.099**	0.105	0.161	0.205	0.417
100	0.144	0.122	0.105	0.099	**0.097**	0.109	0.131	0.310
500	0.156	0.130	0.120	0.114	0.110	0.101	0.099	0.147
1000	0.161	0.148	0.137	0.126	0.118	0.099	0.092	0.088
10000	0.165	0.140	0.120	0.107	0.099	**0.082**	**0.075**	**0.054**

discrete $k = 50, \alpha = 2, m = 10^5, rep = 500$, NRMSE 1-pass

ℓ, T	1	5	20	50	100	500	1000	10000
1	**0.145**	0.172	0.235	0.290	0.345	0.505	0.601	1.063
5	0.174	**0.134**	0.147	0.170	0.202	0.311	0.370	0.636
20	0.243	0.153	0.123	0.126	0.138	0.196	0.232	0.421
50	0.280	0.181	**0.120**	**0.106**	0.104	0.134	0.160	0.282
100	0.343	0.211	0.146	0.116	0.099	0.097	0.107	0.185
500	0.397	0.222	0.141	0.107	0.085	0.046	0.035	0.018
1000	0.384	0.243	0.156	0.110	0.086	0.047	0.036	0.013
10000	0.397	0.231	0.150	0.107	**0.083**	**0.043**	**0.032**	**0.012**

discrete $k = 50, \alpha = 2, m = 10^5, rep = 500$, NRMSE 2-pass

ℓ, T	1	5	20	50	100	500	1000	10000
1	**0.145**	0.172	0.235	0.290	0.345	0.505	0.601	1.063
5	0.165	**0.132**	0.145	0.169	0.201	0.311	0.370	0.636
20	0.205	0.147	0.122	0.125	0.137	0.196	0.232	0.422
50	0.241	0.165	**0.120**	0.104	0.101	0.132	0.158	0.282
100	0.294	0.194	0.140	0.112	0.097	0.094	0.105	0.184
500	0.320	0.202	0.138	0.105	0.082	0.042	0.031	0.016
1000	0.334	0.216	0.146	0.109	0.084	0.041	0.029	0.010
10000	0.326	0.206	0.140	**0.104**	**0.081**	**0.040**	**0.028**	**0.010**

Figure 2: Simulation Results for Discrete SH$_\ell$

continuous $k = 100, \alpha = 1.1, m = 100000, rep = 500$, NRMSE 1-pass

ℓ, T	1	5	20	50	100	500	1000	10000
0.1	0.098	0.118	0.180	0.252	0.326	0.648	0.897	2.399
1	**0.098**	0.108	0.150	0.207	0.267	0.541	0.781	2.006
5	0.109	**0.100**	0.110	0.135	0.170	0.316	0.432	1.135
20	0.114	0.106	0.105	0.112	0.126	0.198	0.252	0.672
50	0.117	0.106	**0.103**	0.105	0.108	0.145	0.179	0.418
100	0.125	0.112	0.103	**0.101**	**0.101**	0.114	0.133	0.285
500	0.141	0.130	0.122	0.119	0.115	0.106	0.103	0.120
1000	0.144	0.133	0.123	0.118	0.112	0.102	0.097	0.083
10000	0.133	0.121	0.115	0.110	0.108	**0.098**	**0.094**	**0.080**

continuous $k = 100, \alpha = 1.1, m = 100000, rep = 500$, NRMSE 2-pass

ℓ, T	1	5	20	50	100	500	1000	10000
0.1	0.098	0.118	0.180	0.252	0.326	0.649	0.897	2.399
1	**0.097**	0.107	0.149	0.206	0.266	0.540	0.780	2.005
5	0.106	**0.100**	0.111	0.135	0.171	0.316	0.433	1.135
20	0.112	0.106	0.104	0.111	0.125	0.197	0.251	0.671
50	0.112	0.106	**0.103**	0.103	0.108	0.144	0.178	0.418
100	0.120	0.111	0.103	**0.101**	**0.100**	0.113	0.131	0.285
500	0.137	0.128	0.121	0.117	0.114	0.105	0.103	0.120
1000	0.138	0.130	0.121	0.115	0.111	0.101	0.097	0.082
10000	0.130	0.119	0.112	0.109	0.106	**0.097**	**0.093**	**0.079**

continuous $k = 100, \alpha = 1.5, m = 100000, rep = 500$, NRMSE 1-pass

ℓ, T	1	5	20	50	100	500	1000	10000
0.1	0.106	0.134	0.198	0.266	0.341	0.558	0.688	1.508
1	**0.103**	0.120	0.168	0.228	0.292	0.478	0.586	1.320
5	0.119	**0.096**	0.113	0.142	0.174	0.301	0.382	0.766
20	0.152	0.115	**0.096**	0.100	0.112	0.176	0.220	0.455
50	0.190	0.136	0.102	**0.092**	0.092	0.121	0.148	0.294
100	0.214	0.152	0.115	0.092	**0.082**	0.078	0.088	0.167
500	0.225	0.169	0.129	0.105	0.089	0.059	0.049	0.025
1000	0.224	0.163	0.122	0.102	0.088	**0.059**	**0.048**	**0.024**
10000	0.230	0.162	0.130	0.108	0.091	0.059	0.049	0.025

continuous $k = 100, \alpha = 1.5, m = 100000, rep = 500$, NRMSE 2-pass

ℓ, T	1	5	20	50	100	500	1000	10000
0.1	0.106	0.134	0.198	0.266	0.341	0.558	0.688	1.508
1	**0.101**	0.118	0.167	0.228	0.292	0.478	0.586	1.320
5	0.113	**0.096**	0.111	0.140	0.172	0.300	0.381	0.766
20	0.142	0.110	**0.095**	0.098	0.110	0.175	0.219	0.454
50	0.168	0.126	0.100	**0.091**	0.091	0.120	0.147	0.294
100	0.191	0.139	0.109	0.091	**0.082**	0.076	0.087	0.167
500	0.203	0.154	0.121	0.102	0.088	0.057	0.045	0.022
1000	0.198	0.146	0.117	0.099	0.086	**0.056**	**0.045**	**0.022**
10000	0.205	0.152	0.122	0.104	0.089	0.057	0.046	0.023

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

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

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

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

Figure 3: Simulation Results for Continuous SH$_\ell$

not attempt to benchmark performance in terms of running time, since computationally, our algorithms are similar to the widely applied distinct sampling or counting algorithms and can easily be tuned and scaled to very large data sets and common platforms.

We generated streams of 10^5 elements with uniform weights. The keys were drawn from a Zipf distribution with parameter $\alpha = 1, 1.1.2, 1.5, 1.8, 2$. This range of Zipf parameters is typical to large data sets and working with them allowed us to finely understand the error dependence on the skew (Zipf with larger α is more skewed and has fewer distinct keys per number of elements). The average number of distinct keys in our simulations, and respective sample sizes we used, was 4.3×10^4 for $\alpha = 1.1$ (used $k = 100$); 1.84×10^4 for $\alpha = 1.2$ (used $k = 100$); 3.04×10^3 for $\alpha = 1.5$ (used $k = 100$); 841 for $\alpha = 1.8$ (used $k = 50$); and 437 for $\alpha = 2$ (used $k = 50$).

For each stream, we computed the exact frequencies of each key for reference in the error computation of the estimates. For a set of sample cap parameters $\ell = 1, 5, 20, 50, 100, 1000, 10000$ (and also $\ell = 0.1$ with continuous samples), we computed discrete and continuous fixed-k SH$_\ell$ samples. Discrete SH$_\ell$ sampling used Algorithm 3 with scoring function (6) and continuous SH$_\ell$ sampling used Algorithm 5.

From each sample, we computed an estimate of the frequency cap statistic $Q(\mathsf{cap}_T, \mathcal{X})$ over all keys, for parameters $T = 1$, 5, 20, 50, 100, 1000, 10000. With discrete SH$_\ell$, we used the estimator of the form (7) and computed estimation coefficients as in Theorem 4.1. With continuous SH$_\ell$, we used the estimator (11) with coefficient function (12), which for cap_T statistics is:

$$\beta(c) = \frac{\min\{T, c\}}{\min\{1, \ell\tau\}} + \tau^{-1} I_{c<T} \ .$$

For each ℓ, T combination, we also computed the estimate that is obtained from 2-pass algorithms (Section 3.1), applied with element scoring (6) for discrete schemes and (9) for continuous schemes. We used the inverse probability estimate $\sum_x \min\{T, w_x\}/\Phi(w_x)$, where $\Phi(w_x)$ is (10) for continuous schemes and as outlined in Section 4 for discrete schemes.

For each of these estimates, we computed the relative and NRMSE errors, averaged over multiple ($rep = 200$ or $rep = 500$) simulations (each using a fresh hash function and randomness). Selected simulation results showing the errors for ℓ, T combinations are provided in Figure 2 for discrete SH$_\ell$ and in Figure 3 for continuous SH$_\ell$. The minimum error for each statistics T across samples ℓ is boldfaced. Additional results are provided in the TR [5].

Discussion: When looking at the parameter ℓ with smallest error for each cap statistics T, we see the diagonal pattern expected from our analysis, where the error is minimized when $\ell \approx T$ and degrades with disparity between T and ℓ. Note that the smallest distinct sampling threshold we had was $\tau \approx 0.001$ (for $\alpha = 1.1$), therefore, our high ℓ values effectively emulated uncapped SH.

Even for these realistic distributions, we observe that a considerable performance gain by using an appropriate sample for our particular cap statistics. We can also see that the sensitivity of the error to the parameter ℓ increases with skew (higher Zipf parameter α). In particular, the ratio of the error to the boldfaced minimum when using a high ℓ sample to estimate distinct counts was up to a factor of 3 whereas the reverse could be 30 fold or more. The increase in error for mid-cap statistics by using the better one of $\ell = 1, \infty$ instead of the minimum was up to 40%. Note however that even this is optimistic, as we measured error on the whole population – on segments with frequency distributions that do not match that of the population, error can be much higher.

Comparing the error of 2-pass versus streaming estimates (both are the same for distinct counts $\ell = 1$ but diverge otherwise), we observe that the benefit of the second pass is limited to 10% and typically lower. This agrees with our CV upper bounds which are only slightly larger for the streaming estimates. This suggests that the choice of scheme should depend on the computational platform.

The $\ell = 1, T = 1$ estimates have NRMSE $\approx 1/\sqrt{k-2}$, this is because the upper bounds for approximate distinct counting are fairly tight [3, 4] as there is no dependence on the frequency distribution. In our simulations, for higher cap values T, the minimum error (over ℓ) was typically much lower than the CV upper bounds. This suggests using adaptive confidence bounds, based on sampled frequencies, rather than relying only on the CV upper bounds.

8. RELATED WORK

There is a large body of work on computing statistics over unaggregated data which we can not hope to cover here. The toolbox includes deterministic algorithms [25], other sampling algorithms [7], and Linear sketches (random linear projections) [22, 1, 21]. Most related to frequency cap statistics are sketches based on p-stable distributions that are designed to estimate frequency moments for $p \in [0, 1]$ [21] and L_p sampling [26, 23]. These techniques do not apply for cap statistics, as there are no appropriate stable distributions for cap functions.

Conclusion

Frequency cap statistics are fundamental to data analysis. We propose a principled and practical sampling solution for scalably and accurately estimating frequency cap statistics over unaggregated data sets. The sample is computed using state proportional to the specified desired sample size and the estimates have error bounds that nearly match those that can be obtained by an optimal weighted sample of the same size that can only be computed over the aggregated view. Our design brings the benefits of approximate distinct counters, which are extensively deployed in the industry, to general frequency cap statistics.

Looking ahead, we would like to apply our framework for sampling unaggregated data sets to other statistics, extend it to support negative updates [15, 6], and understand the theoretical boundaries of the approach.

Acknowledgement: The author would like to thank Kevin Lang for bringing to her attention the use of frequency capping in online advertising and the need for efficient sketches that support it.

9. REFERENCES

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

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

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

[4] E. Cohen. All-distances sketches, revisited: HIP estimators for massive graphs analysis. In *PODS*. ACM, 2014.

[5] E. Cohen. Stream sampling for frequency cap statistics. Technical Report cs.IR/1502.05955, arXiv, 2015. `http://arxiv.org/abs/1502.05955`.

[6] 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.

[7] E. Cohen, N. Duffield, H. Kaplan, C. Lund, and M. Thorup. Composable, scalable, and accurate weight summarization of unaggregated data sets. *Proc. VLDB*, 2(1):431–442, 2009.

[8] 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.

[9] 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.

[10] E. Cohen and H. Kaplan. Tighter estimation using bottom-k sketches. In *Proceedings of the 34th VLDB Conference*, 2008.

[11] 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`.

[12] N. Duffield, M. Thorup, and C. Lund. Priority sampling for estimating arbitrary subset sums. *J. Assoc. Comput. Mach.*, 54(6), 2007.

[13] C. Estan and G. Varghese. New directions in traffic measurement and accounting. In *SIGCOMM*. ACM, 2002.

[14] P. Flajolet and G. N. Martin. Probabilistic counting algorithms for data base applications. *J. Comput. System Sci.*, 31:182–209, 1985.

[15] R. Gemulla, W. Lehner, and P. J. Haas. A dip in the reservoir: Maintaining sample synopses of evolving datasets. In *VLDB*, 2006.

[16] P. Gibbons and Y. Matias. New sampling-based summary statistics for improving approximate query answers. In *SIGMOD*. ACM, 1998.

[17] Google. *Frequency capping: AdWords help*, December 2014. `https://support.google.com/adwords/answer/117579`.

[18] S. Heule, M. Nunkesser, and A. Hall. HyperLogLog in practice: Algorithmic engineering of a state of the art cardinality estimation algorithm. In *EDBT*, 2013.

[19] N. Hohn and D. Veitch. Inverting sampled traffic. In *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pages 222–233, 2003.

[20] 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.

[21] 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.

[22] W. Johnson and J. Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary Math.*, 26, 1984.

[23] H. Jowhari, M. Saglam, and G. Tardos. Tight bounds for Lp samplers, finding duplicates in streams, and related problems. In *PODS*, 2011.

[24] D. E. Knuth. *The Art of Computer Programming, Vol 2, Seminumerical Algorithms*. Addison-Wesley, 1st edition, 1968.

[25] J. Misra and D. Gries. Finding repeated elements. Technical report, Cornell University, 1982.

[26] M. Monemizadeh and D. P. Woodruff. 1-pass relative-error lp-sampling with applications. In *Proc. 21st ACM-SIAM Symposium on Discrete Algorithms*. ACM-SIAM, 2010.

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

[28] M. Osborne. *Facebook Reach and Frequency Buying*, October 2014. `http://citizennet.com/blog/2014/10/01/facebook-reach-and-frequency-buying/`.

[29] B. Rosén. Asymptotic theory for successive sampling with varying probabilities without replacement, I. *The Annals of Mathematical Statistics*, 43(2):373–397, 1972.

[30] Y. Tillé. *Sampling Algorithms*. Springer-Verlag, New York, 2006.