WOR AND p'S: SKETCHES FOR ℓ_p-SAMPLING WITHOUT REPLACEMENT

A PREPRINT

Edith Cohen
Google Research
Tel Aviv University
edith@cohenwang.com

Rasmus Pagh
IT University of Copenhagen
BARC
Google Research
pagh@itu.dk

David P. Woodruff
CMU
dwoodruf@cs.cmu.edu

August 18, 2020

ABSTRACT

Weighted sampling is a fundamental tool in data analysis and machine learning pipelines. Samples are used for efficient estimation of statistics or as sparse representations of the data. When weight distributions are skewed, as is often the case in practice, without-replacement (WOR) sampling is much more effective than with-replacement (WR) sampling: it provides a broader representation and higher accuracy for the same number of samples. We design novel composable sketches for WOR ℓ_p *sampling*, weighted sampling of keys according to a power $p \in [0, 2]$ of their frequency (or for signed data, sum of updates). Our sketches have size that grows only linearly with the sample size. Our design is simple and practical, despite intricate analysis, and based on off-the-shelf use of widely implemented heavy hitters sketches such as `CountSketch`. Our method is the first to provide WOR sampling in the important regime of $p > 1$ and the first to handle signed updates for $p > 0$.

1 Introduction

Weighted random sampling is a fundamental tool that is pervasive in machine learning and data analysis pipelines. A sample serves as a sparse summary of the data and provides efficient estimation of statistics and aggregates.

We consider data $\mathcal{E}$ presented as *elements* in the form of key value pairs $e = (e.\mathsf{key}, e.\mathsf{val})$. We operate with respect to the *aggregated* form of keys and their *frequencies* $\nu_x := \sum_{e|e.\mathsf{key}=x} e.\mathsf{val}$, defined as the sum of values of elements with key x. Examples of such data sets are stochastic gradient updates (keys are parameters and element values are signed and the aggregated form is the combined gradient), search (keys are queries, elements have unit values, and the aggregated form are query-frequency pairs), or training examples for language models (keys are co-occurring terms).

The data is commonly distributed across servers or devices or is streamed and the number of distinct keys is very large. In this scenario it is beneficial to perform computations without explicitly producing a table of key-frequency pairs, as this requires storage or communication that grows linearly with the number of keys. Instead, we use *composable sketches* which are data structures that support (i) *processing a new element* e: Computing a sketch of $\mathcal{E} \cup \{e\}$ from a sketch of $\mathcal{E}$ and e (ii) *merging*: Computing a sketch of $\mathcal{E}_1 \cup \mathcal{E}_2$ from sketches of each $\mathcal{E}_i$ and (iii) are such that the desired output can be produced from the sketch. Composability facilitates parallel, distributed, or streaming computation. We aim to design sketches of small size, because the sketch size determines the storage and communication requirements. For sampling, we aim for the sketch size to be not much larger than the desired sample size.

The case for p's: Aggregation and statistics of functions of the frequencies are essential for resource allocation, planning, and management of large scale systems across application areas. The need for efficiency prompted rich theoretical and applied work on streaming and sketching methods that spanned decades [60, 41, 4, 38, 43, 36, 35, 55, 54]. We study ℓ_p *sampling*, weighted sampling of keys with respect to a power p of their frequency ν_x^p. These samples support estimates of frequency statistics of the general form $\sum_x f(\nu_x) L_x$ for functions of frequency f and constitute

sparse representations of the data. Low powers ($p < 1$) are used to mitigate frequent keys and obtain a better resolution of the tail whereas higher powers ($p > 1$) emphasize more frequent keys. Moreover, recent work suggests that on realistic distributions, ℓ_p samples for $p \in [0, 2]$ provide accurate estimates for a surprisingly broad set of tasks [34].

Sampling is at the heart of stochastic optimization. When training data is distributed [57], sampling can facilitate efficient example selection for training and efficient communication of gradient updates of model parameters. Training examples are commonly weighted by a function of their frequency: Language models [59, 66] use low powers $p < 1$ of frequency to mitigate the impact of frequent examples. More generally, the function of frequency can be adjusted in the course of training to shift focus to rarer and harder examples as training progresses [9]. A sample of examples can be used to produce stochastic gradients or evaluate loss on domains of examples (expressed as frequency statistics). In distributed learning, the communication of dense gradient updates can be a bottleneck, prompting the development of methods that sparsify communication while retaining accuracy [57, 2, 71, 47]. Weighted sampling by the p-th powers of magnitudes complements existing methods that sparsify using heavy hitters (or other methods, e.g., sparsify randomly), provides adjustable emphasis to larger magnitudes, and retains sparsity as updates are composed.

The case for WOR: Weighted sampling is classically considered with (WR) or without (WOR) replacement. We study here the WOR setting. The benefits of WOR sampling were noted in very early work [44, 42, 72] and are becoming more apparent with modern applications and the typical skewed distributions of massive datasets. WOR sampling provides a broader representation and more accurate estimates, with tail norms replacing full norms in error bounds. Figure 1 illustrates these benefits of WOR for Zipfian distributions with ℓ_1 sampling (weighted by frequencies) and ℓ_2 sampling (weighted by the squares of frequencies). We can see that WR samples have a smaller effective sample size than WOR (due to high multiplicity of heavy keys) and that while both WR and WOR well-approximate the frequency distribution on heavy keys, WOR provides a much better approximation of the tail.

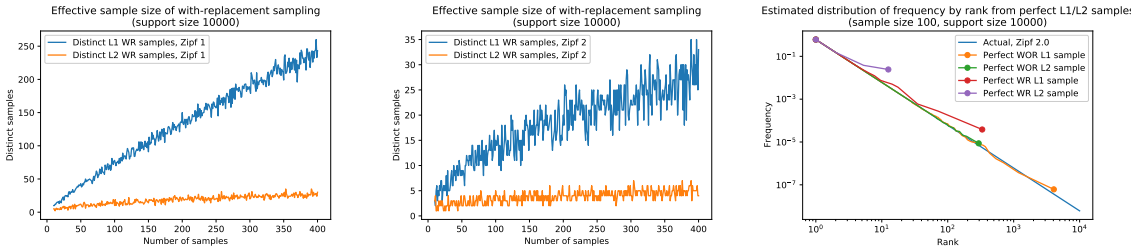

Figure 1: WOR vs WR. Left and middle: Effective vs actual sample size $\mathsf{Zipf}[\alpha = 1]$ and $\mathsf{Zipf}[\alpha = 2]$, with each point reflecting a single sample. Right: Estimates of the frequency distribution $\mathsf{Zipf}[\alpha = 2]$.

Related work. The sampling literature offers many WOR sampling schemes for aggregated data: [68, 72, 14, 69, 64, 37, 25, 26, 23]. A particularly appealing technique is bottom-k (order) sampling, where weights are scaled by random variables and the sample is the set of keys with top-k transformed values [69, 64, 37, 25, 26]. There is also a large body of work on sketches for sampling unaggregated data by functions of frequency. There are two primary approaches. The first approach involves transforming data elements so that a bottom-k sample by function of frequency is converted to an easier problem of finding the top-k keys sorted according to the *maximum* value of an element with the key. This approach yields WOR distinct (ℓ_0) sampling [53], ℓ_1 sampling [41, 22], and sampling with respect to any concave sublinear functions of frequency (including ℓ_p sampling for $p \leq 1$) [20, 24]). These sketches work with non-negative element values but only provide limited support for negative updates [40, 22]. The second approach performs WR ℓ_p sampling for $p \in [0, 2]$ using sketches that are random projections [45, 39, 5, 51, 61, 6, 52, 50]. The methods support signed updates but were not adapted to WOR sampling. For $p > 2$, a classic lower bound [4, 7] establishes that sketches of size polynomial in the number of distinct keys are required for worst case frequency distributions. This task has also been studied in distributed settings [16, 49]; [49] observes the importance of WOR in that setting though does not allow for updates to element values.

Contributions: We present WORp: A method for WOR ℓ_p sampling for $p \in [0, 2]$ via composable sketches of size that grows linearly with the sample size. WORp is simple and practical and uses a bottom-k transform to reduce sampling to *(residual) heavy hitters* (rHH) computation on the transformed data. The technical heart of the paper is establishing that for any set of input frequencies, the keys with top-k transformed frequencies are indeed rHH. In terms of implementation, WORp only requires an off-the-shelf use of popular (and widely-implemented) HH sketches [60, 56, 15, 35, 58, 10]. WORp is the first WOR ℓ_p sampling method (that uses sample-sized sketches) for the regime $p \in (1, 2]$ and the first to fully support negative updates for $p \in (0, 2]$. As a bonus, we include practical

optimizations (that preserve the theoretical guarantees) and perform experiments that demonstrate both the practicality and accuracy of WORp.[1]

In addition to the above, we show that perhaps surprisingly, it is possible to obtain a WOR ℓ_p-sample of a set of k indices, for any $p \in [0,2]$, with variation distance at most $\frac{1}{\mathrm{poly}(n)}$ to a true WOR ℓ_p-sample, and using only $k \cdot \mathrm{poly}(\log n)$ bits of memory. Our variation distance is extremely small, and cannot be detected by any polynomial time algorithm. This makes it applicable in settings for which privacy may be a concern; indeed, this shows that no polynomial time algorithm can learn anything from the sampled output other than what follows from a simulator who outputs a WOR ℓ_p-sample from the actual (variation distance 0) distribution. Finally, for $p \in (0,2)$, we show that the memory of our algorithm is optimal up to an $O(\log^2 \log n)$ factor.

2 Preliminaries

A dataset $\mathcal{E}$ consists of *data elements* that are key value pairs $e = (e.\mathsf{key}, e.\mathsf{val})$. The frequency of a key x, denoted $\nu_x := \sum_{e|e.\mathsf{key}=x} e.\mathsf{val}$, is the sum of values of elements with key x. We use the notation $\boldsymbol{\nu}$ for a vector of frequencies of keys.

For a function f and vector $\boldsymbol{w}$, we denote the vector with entries $f(w_x)$ by $f(\boldsymbol{w})$. In particular, $\boldsymbol{w}^p$ is the vector with entries w_x^p that are the p-th powers of the entries of $\boldsymbol{w}$. For vector $\boldsymbol{w} \in \Re^n$ and index i, we denote by $w_{(i)}$ the value of the entry with the i-th largest magnitude in $\boldsymbol{w}$. We denote by $\mathsf{order}(\boldsymbol{w})$ the permutation of the indices $[n] = \{1, 2, \ldots, n\}$ that corresponds to decreasing order of entries by magnitude. For $k \geq 1$, we denote by $\mathsf{tail}_k(\boldsymbol{w})$ the vector with the k entries with largest magnitudes removed (or replaced with 0).

In the remainder of the section we review ingredients that we build on: bottom-k sampling, implementing a bottom-k transform on unaggregated data, and composable sketch structures for residual heavy hitters (rHH).

2.1 Bottom-k sampling (ppswor and priority)

Bottom-k sampling (also called order sampling [69]) is a family of without-replacement weighted sampling schemes of a set $\{(x, w_x)\}$ of key and weight pairs. The weights (x, w_x) are transformed via independent random maps $w_x^T \leftarrow \frac{w_x}{r_x}$, where $r_x \sim \mathcal{D}$ are i.i.d. from some distribution $\mathcal{D}$. The sample includes the pairs (x, w_x) for keys x that are top-k by transformed magnitudes $|\boldsymbol{w^T}|$ [63, 69, 37, 17, 13, 25, 27][2]. For estimation tasks, we also include a *threshold* $\tau := |w^T_{(k+1)}|$, the $(k+1)$-st largest magnitude of transformed weights. Bottom-k schemes differ by the choice of distribution $\mathcal{D}$. Two popular choices are Probability Proportional to Size WithOut Replacement (ppswor) [69, 17, 27] via the exponential distribution $\mathcal{D} \leftarrow \mathsf{Exp}[1]$ and Priority (Sequential Poisson) sampling [63, 37] via the uniform distribution $\mathcal{D} \leftarrow U[0,1]$. Ppswor is equivalent to a weighted sampling process [68] where keys are drawn successively (without replacement) with probability proportional to their weight. Priority sampling mimics a pure Probability Proportional to Size (pps) sampling, where sampling probabilities are proportional to weights (but truncated to be at most 1).

Estimating statistics from a Bottom-k sample Bottom-k samples provide us with unbiased inverse-probability [44] per-key estimates on $f(w_x)$, where f is a function applied to the weight [3, 26, 21, 19]):

$$\widehat{f(w_x)} := \begin{cases} \frac{f(w_x)}{\Pr_{r\sim\mathcal{D}}[r \leq |w_x|/\tau]} & \text{if } x \in S \\ 0 & \text{if } x \notin S \end{cases} . \tag{1}$$

These estimates can be used to sparsify a vector $f(\boldsymbol{w})$ to k entries or to estimate sum statistics of the general form:

$$\sum_x f(w_x) L_x \tag{2}$$

using the unbiased estimate

$$\sum_x \widehat{f(w_x)} L_x := \sum_x \widehat{f(w_x)} L_x = \sum_{x \in S} \widehat{f(w_x)} L_x \ .$$

[1] Code for the experiments is provided in the following Colab notebook https://colab.research.google.com/drive/1JcNokk_KdQz1gfUarKUcxhnoo9CUL-fZ?usp=sharing

[2] Historically, the term bottom-k is due to analogous use of $1/w_x^T$, but here we find it more convenient to work with "top-k"

The quality of estimates depends on f and L. We measure the quality of these unbiased estimates by the sum over keys of the per-key variance. With both ppswor and priority sampling and $f(w) := w$, the sum is bounded by a respective one for pps with replacement. The per-key variance bound is

$$\mathsf{Var}[\widehat{w_x}] \leq \frac{1}{k-1} w_x \|\boldsymbol{w}\|_1 \tag{3}$$

and the respective sum by $\sum_x \mathsf{Var}[\widehat{w_x}] \leq \frac{1}{k-1}\|\boldsymbol{w}\|_1^2$. This can be tightened to $\mathsf{Var}[\widehat{w_x}] \leq \min\{O(\frac{1}{k}) w_x \|\mathsf{tail}_k(\boldsymbol{w})\|_1, \exp\left(-O(k)\frac{w_x}{\|\mathsf{tail}_k(\boldsymbol{w})\|_1}\right) w_x^2\}$ and respective bound on the sum of $O(\|\mathsf{tail}_k(\boldsymbol{w})\|_1^2/k)$. For skewed distributions, $\|\mathsf{tail}_k(\boldsymbol{w})\|_1^2 \ll \|\boldsymbol{w}\|_1^2$ and hence WOR sampling is beneficial. Conveniently, bottom-k estimates for different keys $x_1 \neq x_2$ have non-positive correlations $\mathsf{Cov}[\widehat{w_{x_1}}, \widehat{w_{x_2}}] \leq 0$, so the variance of sum estimates is at most the respective weighted sum of per-key variance. Note that the per-key variance for a function of weight is $\mathsf{Var}[\widehat{f(w_x)}] = \frac{f(w_x)^2}{w_x^2}\mathsf{Var}[\widehat{w_x}]$. WOR (and WR) estimates are more accurate (in terms of normalized variance sum) when the sampling is weighted by $f(\boldsymbol{w})$.

2.2 Bottom-k sampling by power of frequency

To perform bottom-k sampling of $\boldsymbol{w}^p$ with distribution $\mathcal{D}$, we draw $r_x \sim \mathcal{D}$, transform the weights $w_x^T \leftarrow w_x^p/r_x$, and return the top-$k$ keys in $\boldsymbol{w}^T$. This is equivalent to bottom-k sampling the vector $\boldsymbol{w}$ using the distribution $\mathcal{D}^{1/p}$, that is, draw $r_x \sim \mathcal{D}$, transform the weights

$$w_x^* \leftarrow \frac{w_x}{r_x^{1/p}} \tag{4}$$

and return the top-k keys according to $\boldsymbol{w}^*$. Equivalence is because $(w_x^*)^p = \left(\frac{w_x}{r_x^{1/p}}\right)^p = \frac{w_x^p}{r_x} = w_x^T$ and $f(x) = x^p$ is a monotone increasing and hence $\mathsf{order}(\boldsymbol{w}^*) = \mathsf{order}(\boldsymbol{w}^T)$. We denote the distribution of $\boldsymbol{w}^*$ obtained from the bottom-k transform (4) as p-$\mathcal{D}[\boldsymbol{w}]$ and specifically, p-ppswor$[\boldsymbol{w}]$ when $\mathcal{D} = \mathsf{Exp}[1]$ and p-priority$[\boldsymbol{w}]$ when $\mathcal{D} = U[0,1]$. We use the term p-ppswor for bottom-k sampling by $\mathsf{Exp}^{1/p}$.

The linear transform (4) can be efficiently performed over unaggregated data by using a random hash to represent r_x for keys x and then locally generating an output element for each input element

$$(e.\mathsf{key}, e.\mathsf{val}) \rightarrow (e.\mathsf{key}, e.\mathsf{val}/r_{e.\mathsf{key}}^{1/p}) \tag{5}$$

The task of sampling by p-th power of frequency $\boldsymbol{\nu}^p$ is replaced by the task of top-k by frequency $\nu_x^* := \sum_{e\in\mathcal{E}^*|e.\mathsf{key}=x} e.\mathsf{val}$ on the respective output dataset $\mathcal{E}^*$, noting that $\nu_x^* = \nu_x/r_x^{1/p}$. Therefore, the top-$k$ keys in $\boldsymbol{\nu}^*$ are a bottom-k sample according to $\mathcal{D}$ of $\boldsymbol{\nu}^p$. Note that we can approximate the input frequency ν_x' of a key x from an approximate output frequency $\widehat{\nu_x^*}$ using the hash r_x. Note that relative error is preserved:

$$\nu_x' \leftarrow \widehat{\nu_x^*} r_x^{1/p} \ . \tag{6}$$

This per-element scaling was proposed in the *precision sampling* framework of Andoni et al. [6] and inspired by a technique for frequency moment estimation using stable distributions [45].

Generally, finding the top-k frequencies is a task that requires large sketch structures of size linear in the number of keys. However, [6] showed that when the frequencies are drawn from p-priority$[\boldsymbol{w}]$ (applied to arbitrary $\boldsymbol{w}$) and $p \leq 2$ then the top-1 value is likely to be an ℓ_2 *heavy hitter*. Here we refine the analysis and use the more subtle notion of *residual heavy hitters* [10]. We show that the top-k output frequencies in $\boldsymbol{w}^* \sim p$-ppswor$[\boldsymbol{w}]$ are very likely to be ℓ_q residual heavy hitters (when $q \geq p$) and can be found with a sketch of size $\tilde{O}(k)$.

2.3 Residual heavy hitters (rHH)

An entry in a weight vector $\boldsymbol{w}$ is called an ε- *heavy hitter* if $w_x \geq \varepsilon \sum_y w_y$. A heavy hitter with respect to a function f is defined as a key with $f(w_x) \geq \varepsilon \sum_y f(w_y)$. When $f(w) = w^q$, we refer to a key as an ℓ_q heavy hitter. For $k \geq 1$ and $\psi > 0$, a vector $\boldsymbol{w}$ has (k,ψ) *residual heavy hitters* [10] when the top-k keys are "heavy" with respect to the tail of the frequencies starting at the $(k+1)$-st most frequent key, that is, $\forall i \leq k,\ w_{(i)} \geq \frac{\psi}{k}\|\mathsf{tail}_k(\boldsymbol{w})\|_1$. This is equivalent to $\frac{\|\mathsf{tail}_k(\boldsymbol{w})\|_1}{w_{(k)}} \leq \frac{k}{\psi}$. We say that $\boldsymbol{w}$ has rHH with respect to a function f if $f(\boldsymbol{w})$ has rHH. In particular, $\boldsymbol{w}$ has ℓ_q (k,ψ) rHH if

$$\frac{\|\mathsf{tail}_k(\boldsymbol{w})\|_q^q}{w_{(k)}^q} \leq \frac{k}{\psi} \ . \tag{7}$$

Popular composable HH sketches were adopted to rHH and include (see Table 1): (i) ℓ_1 sketches designed for positive data elements. A deterministic counter-based variety [60, 56, 58] with rHH adaptation [10] and the randomized `CountMin` sketch [35]. (ii) ℓ_2 sketches designed for signed data elements, notably `CountSketch` [15] with rHH analysis in [52]. With these designs, a sketch for ℓ_q (k,ψ)-rHH provides estimates $\widehat{\nu_x}$ for all keys x with error bound:

$$\|\widehat{\boldsymbol{\nu}} - \boldsymbol{\nu}\|_\infty^q \leq \frac{\psi}{k}\|\mathsf{tail}_k(\boldsymbol{\nu})\|_q^q \ . \tag{8}$$

With randomized sketches, the error bound (8) is guaranteed with some probability $1-\delta$. `CountSketch` has the advantages of capturing top keys that are ℓ_2 but not ℓ_1 heavy hitters and supports signed data, but otherwise provides lower accuracy than ℓ_1 sketches for the same sketch size. Methods also vary in supported key domains: counters natively work with key strings whereas randomized sketches work for keys from $[n]$ (see Appendix A for a further discussion). We use these sketches off-the-shelf through the following operations:

- R.`Initialize`(k,ψ,δ): Initialize a sketch structure
- `Merge`(R_1,R_2): Merge two sketches with the same parameters and internal randomization
- R.`Process`(e): process a data element e
- R.`Est`(x): Return an estimate of the frequency of a key x .

Sketch (ℓ_q, sign)	Size	$\|\widehat{\boldsymbol{\nu}} - \boldsymbol{\nu}\|_\infty^q \leq$
`Counters` $(\ell_1, +)$ [10]	$O(\frac{k}{\psi})$	$\frac{\psi}{k}\|\mathsf{tail}_k(\boldsymbol{\nu})\|_1$
`CountSketch` $(\ell_2,\pm)$ [15]	$O(\frac{k}{\psi}\log\frac{n}{\delta})$	$\frac{\psi}{k}\|\mathsf{tail}_k(\boldsymbol{\nu})\|_2^2$

Table 1: Sketches for ℓ_q (k,ψ) rHH.

3 WORp Overview

Our WORp methods apply a p-ppswor transform to data elements (5) and (for $q \geq p$) compute an ℓ_q (k,ψ)-rHH sketch of the output elements. The rHH sketch is used to produce a sample of k keys.

We would like to set ψ to be low enough so that for any input frequencies $\boldsymbol{\nu}$, the top-k keys by transformed frequencies $\boldsymbol{\nu}^* \sim p$-ppswor$[\boldsymbol{\nu}]$ are rHH (satisfy condition (7)) with probability at least $1-\delta$. We refine this desiderata to be conditioned on the permutation of keys in $\mathsf{order}(\boldsymbol{\nu}^*)$. This conditioning turns out not to further constrain ψ and allows us to provide the success probability uniformly for any potential k-sample. Since our sketch size grows inversely with ψ (see Table 1), we want to use the maximum value that works. We will be guided by the following:

$$\Psi_{n,k,\rho=q/p}(\delta) := \sup\left\{\psi \mid \forall \boldsymbol{w}\in\Re^n, \pi\in S^n \Pr_{\boldsymbol{w}^*\sim p\text{-ppswor}[\boldsymbol{w}]\mid \mathsf{order}(\boldsymbol{w}^*)=\pi}\left[k\frac{|w^*_{(k)}|^q}{\|\mathsf{tail}_k(\boldsymbol{w}^*)\|_q^q} \leq \psi\right] \leq \delta\right\}, \tag{9}$$

where S^n denotes the set of permutations of $[n]$. If we set the rHH sketch parameter to $\psi \leftarrow \varepsilon\Psi_{n,k,\rho}$ then using (8), with probability at least $1-\delta$,

$$\|\widehat{\boldsymbol{\nu}^*} - \boldsymbol{\nu}^*\|_\infty^q \leq \frac{\psi}{k}\|\mathsf{tail}_k(\boldsymbol{\nu}^*)\|_q^q = \varepsilon\frac{\Psi_{n,k,\rho}(\lambda)}{k}\|\mathsf{tail}_k(\boldsymbol{\nu}^*)\|_q^q \leq \varepsilon|\nu^*_{(k)}|^q \ . \tag{10}$$

We establish the following lower bounds on $\Psi_{n,k,\rho}(\delta)$:

Theorem 3.1. *There is a universal constant $C>0$ such that for all n, $k>1$, and $\rho = q/p$*

$$\textit{For } \rho = 1\textit{:}\ \Psi_{n,k,\rho}(3e^{-k}) \geq \frac{1}{C\ln\frac{n}{k}} \tag{11}$$

$$\textit{For } \rho > 1\textit{:}\ \Psi_{n,k,\rho}(3e^{-k}) \geq \frac{1}{C}\max\{\rho-1, \frac{1}{\ln\frac{n}{k}})\} \ . \tag{12}$$

To set sketch parameters in implementations, we approximate Ψ using simulations of what we establish to be a "worst case" frequency distribution. For this we use a specification of a "worst-case" set of frequencies as part of the proof of Theorem 3.1 (See Appendix B.1). From simulations we obtain that very small values of $C<2$ suffice for $\delta = 0.01$, $\rho\in\{1,2\}$, and $k\geq 10$.

We analyse a few WORp variants. The first we consider returns an exact p-ppswor sample, including exact frequencies of keys, using two passes. We then consider a variant that returns an approximate p-ppswor sample in a single pass. The two-pass method uses smaller rHH sketches and efficiently works with keys that are arbitrary strings.

We also provide another rHH-based technique that provides a guaranteed very small variation distance on k-tuples in a single pass.

	Sketch size		
sign, p	words	key strings	Pr[success]
$\pm, p < 2$	$O(k \log n)$	$O(k)$	$(1 - \frac{1}{\text{poly}(n)})(1 - 3e^{-k})$
$\pm, p = 2$	$O(k \log^2 n)$	$O(k)$	$(1 - \frac{1}{\text{poly}(n)})(1 - 3e^{-k})$
$+, p < 1$	$O(k)$	$O(k)$	$1 - 3e^{-k}$
$+, p = 1$	$O(k \log n)$	$O(k)$	$1 - 3e^{-k}$

Table 2: Two-pass ppswor sampling of k keys according to $\boldsymbol{\nu}^p$. Sketch size (memory words and number of stored key strings). For $p \in (0, 2]$ and signed ($\pm$) or positive ($+$) value elements.

4 Two-pass WORp

We design a two-pass method for ppswor sampling according to $\boldsymbol{\nu}^p$ for $p \in (0, 2]$ (Equivalently, a p-ppswor sample according to $\boldsymbol{\nu}$):

- **Pass I:** We compute an ℓ_q $(k+1, \psi)$-rHH sketch R of the transformed data elements

$$(\texttt{KeyHash}(e.\mathsf{key}), e.\mathsf{val}/r_{e.\mathsf{key}}^{1/p}) \ . \tag{13}$$

A hash $\texttt{KeyHash} \rightarrow [n]$ is applied when keys have a large or non-integer domain to facilitate use of `CountSketch` or reduce storage with `Counters`. We set $\psi \leftarrow \frac{1}{3^q}\Psi_{n,k,\rho}(\delta)$.

- **Pass II:** We collect key strings x (if needed) and corresponding exact frequencies ν_x for keys with the Bk largest $|\widehat{\nu_x^*}|$, where B is a constant (see below) and $\widehat{\nu_x^*} := R.\texttt{Est}[\texttt{KeyHash}(x)]$ are the estimates of ν_x^* provided by R. For this purpose we use a composable top-(Bk) sketch structure T. The size of T is dominated by storing Bk key strings.

- **Producing a p-ppswor sample from T:** Compute exact transformed frequencies $\nu_x^* \leftarrow \nu_x r_x^{1/p}$ for stored keys $x \in T$. Our sample is the set of key frequency pairs (x, ν_x) for the top-k stored keys by ν_x^*. The threshold τ is the $(k+1)^{\text{th}}$ largest ν_x^* over stored keys.

- **Estimation:** We compute per-key estimates as in (1): Plugging in $\mathcal{D} = \mathsf{Exp}[1]^{1/p}$ for p-ppswor, we have $\widehat{f(\nu_x)} := 0$ for $x \notin S$ and for $x \in S$ is $\widehat{f(\nu_x)} := \frac{f(\nu_x)}{1-\exp\left(-(\frac{\nu_x}{\tau})^p\right)}$.

We establish that the method returns the p-ppswor sample with probability at least $1 - \delta$, propose practical optimizations, and analyze the sketch size:

Theorem 4.1. *The 2-pass method returns a p-ppswor sample of size k according to $\boldsymbol{\nu}$ with success probability and composable sketch sizes as detailed in Table 2. The success probability is defined to be that of returning the exact top-k keys by transformed frequencies. The bound applies even when conditioned on the top-k being a particular k-tuple.*

Proof. From (10), the estimates $\widehat{\nu_x^*} = R.\texttt{Est}[\texttt{KeyHash}(x)]$ of ν_x^* are such that:

$$\Pr\left[\forall x \in \{e.\mathsf{key} \mid e \in \mathcal{E}\}, |\widehat{\nu_x^*} - \nu_x^*| \leq \frac{1}{3}|\nu_{(k+1)}^*|\right] \geq 1 - \delta \ . \tag{14}$$

We set B in the second pass so that the following holds:

$$\text{The top-}(k+1)\text{ keys by } \boldsymbol{\nu}^* \text{ are a subset of the top-}(B(k+1))\text{ keys by } \widehat{\boldsymbol{\nu}^*}. \tag{15}$$

Note that for any frequency distribution with rHH, it suffices to store $O(k/\psi)$ keys to have (15). We establish that for our particular distributions, a constant B suffices. For that we used the following:

Lemma 4.1. *A sufficient condition for property (15) is that $|\nu_{(B(k+1))}^*| \leq \frac{1}{3}|\nu_{((k+1))}^*|$.*

Proof. Note that $|\widehat{\nu_x^*}| \geq \frac{2}{3}|\nu_{(k+1)}^*|$ for keys that are top-$(k+1)$ by $\boldsymbol{\nu}^*$ and $|\widehat{\nu^*}_{(B(k+1))}| \leq |\nu_{(B(k+1))}^*| + \frac{1}{3}|\nu^*(k+1)|$. Hence $|\widehat{\nu_x^*}| \geq |\widehat{\nu^*}_{(B(k+1)}|$ for all keys that are top-$(k+1)$ by $\boldsymbol{\nu}^*$. □

We then use Lemma E.1 to express a "worst-case" distribution for the ratio $\nu^*_{B(k+1)}/\nu^*_(k+1)$ and use the latter (using Corollary D.2) to show that the setting of $\Psi(\delta)$ according to our proof of Theorem 3.1 (Appendix B-D) implies that the conditions that guarantee the rHH property will also imply a ratio of at most $1/3$ with a constant B.

Correctness for the final sample follows from property (15) : T storing the top-$(k+1)$ keys in the data according to $\boldsymbol{\nu}^*$. To conclude the proof of Theorem 4.1 we need to specify the rHH sketch structure we use. From Theorem 3.1 we obtain a lower bound on $\Psi_{n,k,\rho}$ for $\delta = 3e^{-k}$ and we use it to set ψ. For our rHH sketch we use `CountSketch` ($q = 2$ and supports signed values) or `Counters` ($q = 1$ and positive values). The top two lines in Table 2 are for `CountSketch` and the next two lines are for `Counters`. The rHH sketch sizes follow from ψ and Table 1. □

4.1 Practical optimizations

We propose practical optimizations that improve efficiency without compromising quality guarantees.

The first optimization allows us to store $k' \ll B(k+1)$ keys in the second pass: We always store the top-$(k+1)$ keys by $\widehat{\boldsymbol{\nu}^*}$ but beyond that only store keys if they satisfy

$$\widehat{\nu^*}_x \geq \frac{1}{2}\widehat{\nu^*}_{(k+1)} \ , \tag{16}$$

where $\boldsymbol{\nu}^*$ is with respect to data elements processed by the current sketch. We establish correctness:

Lemma 4.2. *1. There is a composable structure that only stores keys that satisfy (16) and collects exact frequencies for these keys.*

2. If (14) holds, the top-$(k+1)$ keys by $\boldsymbol{\nu}^*$ satisfy (16) (and hence will be stored in the sketch).

Proof. (i) The structure is a slight modification of a top-k' structure. Since $\widehat{\nu^*}_{(k+1)}$ can only increase as more distinct keys are processed, the condition (16) only becomes more stringent as we merge sketches and process elements. Therefore if a key satisfies the condition at some point it would have satisfied the condition when elements with the key were previously processed and therefore we can collect exact frequencies.

(ii) From the uniform approximation (14), we have $\widehat{\nu^*}_{(k+1)} \leq \frac{4}{3}\nu^*_{(k+1)}$. Let x be the $(k+1)$-th key by $|\boldsymbol{\nu}^*|$. Its estimated transformed frequency is at at least $\widehat{\nu^*_x} \geq \frac{2}{3}\nu^*_{(k+1)} \geq \frac{2}{3}\cdot\frac{3}{4}\widehat{\nu^*}_{(k+1)} = \frac{1}{2}\widehat{\nu^*}_{(k+1)}$. Therefore, if we store all keys x with $\widehat{\nu^*_x} \geq \frac{1}{2}\widehat{\nu^*}_{(k+1)}$ we store the top-$(k+1)$ keys by ν^*_x. □

A second optimization allows us to extract a larger effective sample from the sketch with $k' \geq k$ keys. This can be done when we can certify that the top-k' keys by $\boldsymbol{\nu}^*$ in the transformed data are stored in the sketch T. Using a larger sample is helpful as it can only improve (in expectation) estimation quality (see e.g., [28, 31]). To extend this, we compute the uniform error bound $\nu^*_{(k+1)}/3$ (available because the top-$(k+1)$ keys by ν^* are stored). Let $L \leftarrow \min_{x\in T}\widehat{\nu^*_x}$. We obtain that any key x in the dataset with $\nu^*_x \geq L + \nu^*_{(k+1)}/3$ must be stored in T. Our final sample contains these keys with the minimum ν^*_x in the set used as the threshold τ.

5 One-pass WORp

Our 1-pass WORp yields a sample of size k that approximates a p-ppswor sample of the same size and provides similar guarantees on estimation quality.

- **Sketch:** For $q \geq p$ and $\varepsilon \in (0, \frac{1}{3}]$ Compute an ℓ_q $(k+1, \psi)$-rHH sketch R of the transformed data elements (5) where $\psi \leftarrow \varepsilon^q \Psi_{n,k+1,\rho}$.
- **Produce a sample:** Our sample S includes the keys with top-k estimated transformed frequencies $\widehat{\nu^*_x} := R.\texttt{Est}[x]$. For each key $x \in S$ we include (x, ν'_x), where the approximate frequency $\nu'_x \leftarrow \widehat{\nu^*_x} r_x^{1/p}$ is according to (6). We include with the sample the threshold $\tau \leftarrow \widehat{\nu^*}_{(k+1)}$.
- **Estimation:** We treat the sample as a p-ppswor sample and compute per-key estimates as in (1), substituting approximate frequencies ν'_x for actual frequencies ν_x of sampled keys and the 1-pass threshold $\widehat{\nu^*}_{(k+1)}$ for the exact

ℓ_p	α	p'	perfect WR	perfect WOR	1-pass WORp	2-pass WORp
ℓ_2	Zipf[2]	ν^3	1.16e-04	2.09e-11	1.06e-03	2.08e-11
ℓ_2	Zipf[2]	ν^2	7.96e-05	1.26e-07	1.14e-02	1.25e-07
ℓ_1	Zipf[2]	ν	9.51e-03	1.60e-03	2.79e-02	1.60e-03
ℓ_1	Zipf[1]	ν^3	3.59e-01	5.73e-03	5.14e-03	5.72e-03
ℓ_1	Zipf[2]	ν^3	3.45e-04	7.34e-10	5.11e-05	7.38e-10

Table 3: NRMSE on estimates of frequency moments on statistics of the form $\|\boldsymbol{\nu}\|_{p'}^{p'}$ from ℓ_p samples ($p = 1, 2$). Zipf[α] distributions with support size $n = 10^4$, $k = 100$ samples, averaged over 100 runs. `CountSketch` of size $k \times 31$

$\nu^*_{(k+1)}$. The estimate is $\widehat{f(\nu_x)} := 0$ for $x \notin S$ and for $x \in S$ is:

$$\widehat{f(\nu_x)} := \frac{f(\nu'_x)}{1 - \exp\left(-(\frac{\nu'_x}{\widehat{\nu^*}_{(k+1)}})^p\right)} = \frac{f(\widehat{\nu^*_x} r_x^{1/p})}{1 - \exp\left(-r_x(\frac{\widehat{\nu^*_x}}{\widehat{\nu^*}_{(k+1)}})^p\right)} \tag{17}$$

We relate the quality of the estimates to those of a perfect p-ppswor. Since our 1-pass estimates are biased (unlike the unbiased perfect p-ppswor), we consider both bias and variance. The proof is provided in Appendix G.

Theorem 5.1. *Let $f(w)$ be such that $|f((1+\varepsilon)w) - f(w)| \leq c\varepsilon f(w)$ for some $c > 0$ and $\varepsilon \in [0, 1/2]$. Let $\widehat{f(\nu_x)}$ be per-key estimates obtained with a one-pass WORp sample and let $\widehat{f(\nu_x)}'$ be the respective estimates obtained with a (perfect) p-ppswor sample. Then $\mathsf{Bias}[\widehat{f(\nu_x)}] \leq O(\varepsilon) f(\nu_x)$ and $\mathsf{MSE}[\widehat{f(\nu_x)}] \leq (1 + O(\varepsilon))\mathsf{Var}[\widehat{f(\nu_x)}'] + O(\epsilon) f(\nu_x)^2$.*

Note that the assumption on f holds for $f(w) = w^p$ with $c = (1.5)^p$. Also note that the bias bound implies a respective contribution to the relative error of $O(\varepsilon)$ on all sum estimates.

6 One-pass Total Variation Distance Guarantee

We provide another 1-pass method, based on the combined use of rHH and known WR perfect ℓ_p sampling sketches [50] to select a k-tuple with a polynomially small total variation (TV) distance from the k-tuple distribution of a perfect p-ppswor. The method uses $O(k)$ (for variation distance $2^{-\Theta(k)}$, and $O(k \log n)$ for variation distance $1/n^C$ for an arbitrarily large constant $C > 0$) perfect samplers (each providing a single WR sample) and an rHH sketch. The perfect samplers are processed in sequence with prior selections "subtracted" from the linear sketch (using approximate frequencies provided by the rHH sketch) to uncover fresh samples. As with WORp, exact frequencies of sampled keys can be obtained in a second pass or approximated using larger sketches in a single pass. Details are provided in Appendix F.

Theorem 6.1. *There is a 1-pass method via composable sketches of size $O(k \operatorname{polylog}(n))$ that returns a k-tuple of keys such that the total variation distance from the k-tuples produced by a perfect p-ppswor sample is at most $1/n^C$ for an arbitrarily large constant $C > 0$. The method applies to keys from a domain $[n]$, and signed values with magnitudes and intermediate frequencies that are polynomial in n.*

We also show in Appendix F that our sketches in Theorem 6.1 use $O(k \cdot \log^2 n (\log \log n)^2)$ bits of memory for $0 < p < 2$, and we prove a matching lower bound on the memory required of any algorithm achieving this guarantee, up to a $(\log \log n)^2$ factor. For $p = 2$ we also show they are of optimal size, up to an $O(\log n)$ factor.

7 Experiments

We implemented 2-pass and 1-pass WORp in Python using `CountSketch` as our rHH sketch. Figure 2 reports estimates of the rank-frequency distribution obtained with 1-pass and 2-pass WORp and perfect WOR (p-ppswor) and perfect WR samples (shown for reference). For best comparison, all WOR methods use the same randomization of the p-ppswor transform. Table 3 reports normalized root averaged squared errors (NRMSE) on example statistics. As expected, 2-pass WORp and perfect 2-ppswor are similar and WR ℓ_2 samples are less accurate with larger skew or on the tail. Note that current state of the art sketching methods are not more efficient for WR sampling than for WOR sampling, and estimation quality with perfect methods is only shown for reference. We can also see that 1-pass WORp performs well, although it requires more accuracy (lager sketch size) since it works with estimated weights (reported results are with fixed `CountSketch` size of $k \times 31$ for all methods).

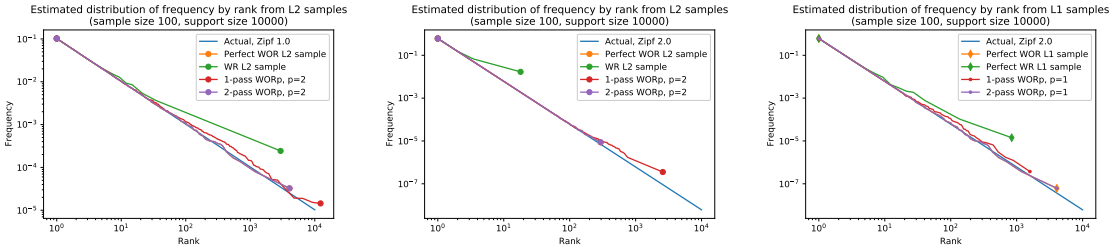

Figure 2: Estimates of the rank-frequency distribution of Zipf[1] and Zipf[2]. Using WORp 1-pass, WORp 2-pass with `CountSketch` (matrix $k \times 31$), perfect WOR, and perfect WR. Estimates from a (representative) single sample of size $k = 100$. Left and Center: ℓ_2 sampling. Right: ℓ_1 sampling.

Conclusion

We present novel composable sketches for without-replacement (WOR) ℓ_p sampling, based on "reducing" the sampling problem to a heavy hitters (HH) problem. The reduction, which is simple and practical, allows us to use existing implementations of HH sketches to perform WOR sampling. Moreover, streaming HH sketches that support time decay (for example, sliding windows [8]) provide a respective time-decay variant of sampling. We present two approaches, WORp, based on a bottom-k transform, and another technique based on "perfect" with-replacement sampling sketches, which provide 1-pass WOR samples with negligible variation distance to a true sample. Our methods open the door for a wide range of future applications. Our WORp schemes produce bottom-k samples (aka bottom-k sketches) with respect to a specified randomization r_x over the support (with 1-pass WORp we obtain approximate bottom-k samples). Samples of different datasets or different p values or different time-decay functions that are generated with the same r_x are *coordinated* [12, 70, 65, 17, 32, 13]. Coordination is a desirable and powerful property: Samples are locally sensitivity (LSH) and change little with small changes in the dataset [12, 70, 46, 33, 19]. This LSH property allows for a compact representation of multiple samples, efficient updating, and sketch-based similarity searches. Moreover, coordinated samples (sketches) facilitate powerful estimators for multi-set statistics and similarity measures such as weighted Jaccard similarity, min or max sums, and one-sided distance norms [17, 13, 28, 31, 29, 30, 18].

Acknowledgments: D. Woodruff would like to thank partial support from the Office of Naval Research (ONR) grant N00014-18-1-2562, and the National Science Foundation (NSF) under Grant No. CCF-1815840.

References

[1] Pankaj K. Agarwal, Graham Cormode, Zengfeng Huang, Jeff M. Phillips, Zhewei Wei, and Ke Yi. Mergeable summaries. *ACM Trans. Database Syst.*, 38(4):26:1–26:28, 2013.

[2] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. Qsgd: Communication-efficient sgd via gradient quantization and encoding. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 1709–1720. Curran Associates, Inc., 2017.

[3] N. Alon, N. Duffield, M. Thorup, and C. Lund. Estimating arbitrary subset sums with few probes. In *Proceedings of the 24th ACM Symposium on Principles of Database Systems*, pages 317–325, 2005.

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

[5] Alexandr Andoni, Khanh Do Ba, Piotr Indyk, and David P. Woodruff. Efficient sketches for earth-mover distance, with applications. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, October 25-27, 2009, Atlanta, Georgia, USA*, pages 324–330. IEEE Computer Society, 2009.

[6] Alexandr Andoni, Robert Krauthgamer, and Krzysztof Onak. Streaming algorithms via precision sampling. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, 2011.

[7] Ziv Bar-Yossef, Thathachar S Jayram, Ravi Kumar, and D Sivakumar. An information statistics approach to data stream and communication complexity. *Journal of Computer and System Sciences*, 68(4):702–732, 2004.

[8] R. Ben-Basat, G. Einziger, R. Friedman, and Y. Kassner. Heavy hitters in streams and sliding windows. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, 2016.

[9] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *ICML*, 2009.

[10] Radu Berinde, Graham Cormode, Piotr Indyk, and Martin J. Strauss. Space-optimal heavy hitters with strong error bounds. In *Proceedings of the Twenty-Eighth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '09. Association for Computing Machinery, 2009.

[11] Vladimir Braverman, Stephen R. Chestnut, Nikita Ivkin, Jelani Nelson, Zhengyu Wang, and David P. Woodruff. Bptree: An l2 heavy hitters algorithm using constant memory. In *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2017, Chicago, IL, USA, May 14-19, 2017*, 2017.

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

[13] A. Z. Broder. On the resemblance and containment of documents. In *Proceedings of the Compression and Complexity of Sequences*, pages 21–29. IEEE, 1997.

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

[15] M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams. In *Proceedings of the International Colloquium on Automata, Languages and Programming (ICALP)*, pages 693–703, 2002.

[16] Yung-Yu Chung, Srikanta Tirthapura, and David P. Woodruff. A simple message-optimal algorithm for random sampling from a distributed stream. *IEEE Trans. Knowl. Data Eng.*, 28(6):1356–1368, 2016.

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

[18] E. Cohen. Distance queries from sampled data: Accurate and efficient. In *KDD*. ACM, 2014. full version: `http://arxiv.org/abs/1203.4903`.

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

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

[21] E. Cohen. Stream sampling framework and application for frequency cap statistics. *ACM Trans. Algorithms*, 14(4):52:1–52:40, 2018. preliminary version published in KDD 2015. arXiv:`http://arxiv.org/abs/1502.05955`.

[22] 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.

[23] 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.

[24] E. Cohen and O. Geri. Sampling sketches for concave sublinear functions of frequencies. In *NeurIPS*, 2019.

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

[26] E. Cohen and H. Kaplan. Sketch-based estimation of subpopulation-weight. Technical Report 802.3448, CORR, 2008.

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

[28] E. Cohen and H. Kaplan. Leveraging discarded samples for tighter estimation of multiple-set aggregates. In *ACM SIGMETRICS*, 2009.

[29] E. Cohen and H. Kaplan. Get the most out of your sample: Optimal unbiased estimators using partial information. In *Proc. of the 2011 ACM Symp. on Principles of Database Systems (PODS 2011)*. ACM, 2011. full version: `http://arxiv.org/abs/1203.4903`.

[30] E. Cohen and H. Kaplan. What you can do with coordinated samples. In *The 17th. International Workshop on Randomization and Computation (RANDOM)*, 2013. full version: `http://arxiv.org/abs/1206.5637`.

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

[32] E. Cohen, Y.-M. Wang, and G. Suri. When piecewise determinism is almost true. In *Proc. Pacific Rim International Symposium on Fault-Tolerant Systems*, pages 66–71, December 1995.

[33] Edith Cohen, Graham Cormode, Nick Duffield, and Carsten Lund. On the tradeoff between stability and fit. *ACM Trans. Algorithms*, 13(1), 2016.

[34] Edith Cohen, Ofir Geri, and Rasmus Pagh. Composable sketches for functions of frequencies: Beyond the worst case. In *ICML*, 2020.

[35] G. Cormode and S. Muthukrishnan. An improved data stream summary: The count-min sketch and its applications. *J. Algorithms*, 55(1), 2005.

[36] N. Duffield, C. Lund, and M. Thorup. Estimating flow distributions from sampled flow statistics. In *Proceedings of the ACM SIGCOMM'03 Conference*, pages 325–336, 2003.

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

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

[39] Gereon Frahling, Piotr Indyk, and Christian Sohler. Sampling in dynamic data streams and applications. *Int. J. Comput. Geom. Appl.*, 18(1/2):3–28, 2008.

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

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

[42] H. O. Hartley and J. N. K. Rao. Sampling with unequal probabilities and without replacement. *Annals of Mathematical Statistics*, 33(2), 1962.

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

[44] 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.

[45] 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.

[46] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proc. 30th Annual ACM Symposium on Theory of Computing*, pages 604–613. ACM, 1998.

[47] Nikita Ivkin, Daniel Rothchild, Enayat Ullah, Vladimir Braverman, Ion Stoica, and Raman Arora. Communication-efficient distributed SGD with sketching. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, 2019.

[48] Svante Janson. Tail bounds for sums of geometrics and exponential variables. https://http://www2.math.uu.se/~svante/papers/sj328.pdf, 2017.

[49] Rajesh Jayaram, Gokarna Sharma, Srikanta Tirthapura, and David P. Woodruff. Weighted reservoir sampling from distributed streams. In Dan Suciu, Sebastian Skritek, and Christoph Koch, editors, *Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019*, pages 218–235. ACM, 2019.

[50] Rajesh Jayaram and David P. Woodruff. Perfect lp sampling in a data stream. In *FOCS*, 2018.

[51] T. S. Jayram and David P. Woodruff. The data stream space complexity of cascaded norms. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, October 25-27, 2009, Atlanta, Georgia, USA*, pages 765–774. IEEE Computer Society, 2009.

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

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

[54] Zaoxing Liu, Ran Ben-Basat, Gil Einziger, Yaron Kassner, Vladimir Braverman, Roy Friedman, and Vyas Sekar. Nitrosketch: robust and general sketch-based monitoring in software switches. In *Proceedings of the ACM Special Interest Group on Data Communication, SIGCOMM 2019, Beijing, China, August 19-23, 2019*, pages 334–350, 2019.

[55] Zaoxing Liu, Antonis Manousis, Gregory Vorsanger, Vyas Sekar, and Vladimir Braverman. One sketch to rule them all: Rethinking network flow monitoring with univmon. In *SIGCOMM*, 2016.

[56] G. Manku and R. Motwani. Approximate frequency counts over data streams. In *International Conference on Very Large Databases (VLDB)*, pages 346–357, 2002.

[57] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*. PMLR, 2017.

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

[59] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013.

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

[61] M. Monemizadeh and D. P. Woodruff. 1-pass relative-error $\mathrm{l_p}$-sampling with applications. In *Proc. 21st ACM-SIAM Symposium on Discrete Algorithms*. ACM-SIAM, 2010.

[62] Rajeev Motwani and Prabhakar Raghavan. Randomized algorithms. In *Algorithms and Theory of Computation Handbook*. 1999.

[63] E. Ohlsson. Sequential poisson sampling from a business register and its application to the swedish consumer price index. Technical Report 6, Statistics Sweden, 1990.

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

[65] E. Ohlsson. Coordination of pps samples over time. In *The 2nd International Conference on Establishment Surveys*, pages 255–264. American Statistical Association, 2000.

[66] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *EMNLP*, 2014.

[67] Eric Price. Efficient sketches for the set query problem. In Dana Randall, editor, *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 41–56. SIAM, 2011.

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

[69] B. Rosén. Asymptotic theory for order sampling. *J. Statistical Planning and Inference*, 62(2):135–158, 1997.

[70] P. J. Saavedra. Fixed sample size pps approximations with a permanent random number. In *Proc. of the Section on Survey Research Methods*, pages 697–700, Alexandria, VA, 1995. American Statistical Association.

[71] Sebastian U. Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified sgd with memory. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18. Curran Associates Inc., 2018.

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

A Properties of rHH sketches

Sketches for ℓ_1 heavy hitters on datasets with positive values: These include the deterministic counter-based Misra Gries [60, 1], Lossy Counting [56], and Space Saving [58] and the randomized Count-Min Sketch [35]. A sketch of size $O(\varepsilon^{-1})$ provides frequency estimates with absolute error at most $\varepsilon\|\boldsymbol{\nu}\|_1$. Berinde et al. [10] provide a counter-based sketch of size $O(k/\psi)$ that provides absolute error at most $\frac{\psi}{k}\|\mathsf{tail}_k(\boldsymbol{\nu})\|_1$.

Sketches for ℓ_2 heavy hitters on datasets with signed values: Pioneered by `CountSketch` [15]: A sketch of size $O(\varepsilon^{-1}\log\frac{n}{\delta})$ provides with confidence $1-\delta$ estimates with error bound $\varepsilon\|\boldsymbol{\nu}\|_2^2$ for squared frequencies. For rHH, a `CountSketch` of size $O(\frac{k}{\psi}\log\frac{n}{\delta})$ provides estimates for all squared frequencies with absolute error at most $\frac{\psi}{k}\|\mathsf{tail}_k(\boldsymbol{\nu})\|_2^2$. These bounds were further refined in [52] for ℓ_p rHH. The dependence on $\log n$ was replaced by $1/\psi$ in [11] for insertion only streams. Unlike the case for counter-based sketches, the estimates produced by `CountSketch` are unbiased, a helpful property for estimation tasks.

Obtaining the rHH keys Keys can be arbitrary strings (search queries, URLs, terms) or integers from a domain $[n]$ (parameters in an ML model). Keys in the form of strings can be handled by hashing them to a domain $[n]$ but we discuss applications that require the sketch to return rHH keys in their string form. Counter-based sketches store explicitly $O(k/\psi)$ keys. The stored keys can be arbitrary strings. The estimates are positive for stored keys and 0 for other keys. The rHH keys are contained in the stored keys. The randomized rHH sketches (`CountSketch` and `CountMin`) are designed for keys that are integers in $[n]$. The bare sketch does not explicitly store keys. The rHH keys can be recovered by enumerating over $[n]$ and retaining the keys with largest estimates. Alternatively, when streaming

(performing one sequential pass over elements) we can maintain an auxiliary structure that holds key strings with current top-k estimates [15].

With general composable sketches, key strings can be handled using an auxiliary structure that increases the sketch size by a factor linear in string length. This is inefficient compared with sketches that natively store the string. Alternatively, a two-pass method can be used with the first pass computing an rHH sketch for a hashed numeric representation and a second pass is used to obtain the key strings of hashed representations with high estimates.

Recovering (approximate) frequencies of rHH keys For our application, we would need to have approximate or exact frequencies of rHH keys. The estimates provided by a (k, ψ) rHH sketch provide absolute error (statistical) guarantees (see Table 1). One approach is to recover exact frequencies in a second pass. We can also obtain more accurate estimates (of relative error at most ϵ) by using a $(k, \epsilon\psi)$ rHH sketch.

Testing for failure Recall that the dataset may not have (k, ψ) rHH. We can test if one of the k largest estimated frequencies to the pth power is below the specified error bound of $\geq \frac{\psi}{k}\|\mathsf{tail}_k(\boldsymbol{\nu})\|_p^p$. If so, we declare "failure."

B Overview of the proof of Theorem 3.1

For a vector $\boldsymbol{w} \in \Re^n$, permutation $\pi \in S^n$, and $p > 0$, let the random variable $\boldsymbol{w}^* \sim p\text{-ppswor}[\boldsymbol{w}] \mid \pi$ be a p-ppswor transform (4) of $\boldsymbol{w}$ conditioned on the event $\mathsf{order}(\boldsymbol{w}^*) = \pi$. For $q > p$ and $k > 1$, we define the following distribution:

$$F_{\boldsymbol{w},p,q,k} \mid \pi := \frac{\|\mathsf{tail}_k(\boldsymbol{w}^*)\|_q^q}{(w^*_{(k)})^q} \ . \tag{18}$$

Note that for any $\boldsymbol{w} \in \Re^n$ and $\pi \in S^n$,

$$\Pr_{\boldsymbol{w}^* \sim p\text{-ppswor}[\boldsymbol{w}] \mid \mathsf{order}(\boldsymbol{w}^*)=\pi}\left[k\frac{|w^*_{(k)}|^q}{\|\mathsf{tail}_k(\boldsymbol{w}^*)\|_q^q} \leq \psi\right] = \Pr_{z \sim F_{\boldsymbol{w},p,q,k}|\pi}\left[z \leq \frac{k}{\psi}\right] \tag{19}$$

Therefore tail bounds on F that hold for any $\boldsymbol{w} \in \Re^n$ and $\pi \in S^n$ can be used to establish the claim.

We now define another distribution that does not depend on $\boldsymbol{w}$ and π:

Definition B.1. *For $1 \leq k \leq n$ and $\rho \geq 1$ we define a distribution $R_{n,k,\rho}$ as follows.*

$$R_{k,n,\rho} := \sum_{i=k+1}^{n} \frac{\left(\sum_{j=1}^{k} Z_j\right)^\rho}{\left(\sum_{j=1}^{i} Z_j\right)^\rho} \ ,$$

where $Z_i \sim \mathsf{Exp}[1]$ $i \in [n]$ are i.i.d.

The proof of Theorem 3.1 will follow using the following two components:

(i) We show (Section C) that for any $\boldsymbol{w} \in \Re^n$ and permutation $\pi \in S^n$,

$$F_{\boldsymbol{w},p,q,k}|\pi \preceq R_{k,n,\rho=(q/p)} \ ,$$

where the relation $\preceq$ corresponds here to statistical domination of distributions.

(ii) We establish (Section D) tail bounds on $R_{k,n,\rho=(q/p)}$.

Because of domination, the tails bounds on $R_{k,n,\rho=(q/p)}$ provide corresponding tail bound for $F_{\boldsymbol{w},p,q,k}|\pi$ for any $\boldsymbol{w} \in \Re^n$ and $\pi \in S^n$. Together with (19), we use the tail bounds to conclude the proof of Theorem 3.1.

Moreover, the domination relation is tight in the sense that for some $\boldsymbol{w}$ and π, $F_{\boldsymbol{w},p,q,k}|\pi$ is very close to $R_{k,n,q/p}$: For distributions with k keys with relative frequency ϵ and π that has these keys in the first k (as $\epsilon \to 0$), or for uniform distributions with $n \gg k$, $F_{\boldsymbol{w},p,q,k}|\pi$ (as n grows).

The tail bounds (and hence the claim of Theorem 3.1) also hold without the condition on π.

Lemma B.2. *The tail bounds also hold for the unconditional distribution $F_{\boldsymbol{w},p,q,k}$.*

Proof. The distribution $F_{\boldsymbol{w},p,q,k}$ is a convex combination of distributions $F_{\boldsymbol{w},p,q,k}|\pi$. Specifically, for each permutation π let p_π be the probability that we obtain this permutation with successive weighted sampling with replacement. Then

$$F_{\boldsymbol{w},p,q,k} = \sum_\pi p_\pi F_{\boldsymbol{w},p,q,k}|\pi \ . \tag{20}$$

Since tail bounds hold for each term, they hold for the combination. □

B.1 Approximating ψ by simulations

$\Psi_{k,n,\rho}(\delta)$ is the solution of the following for ψ:

$$\Pr_{z\sim R_{k,n,\rho}}[z \geq k/\psi] = \delta \ . \tag{21}$$

We can approximate $\Psi_{k,n,\rho}(\delta)$ by computing i.i.d. $z_i \sim R_{k,n,\rho}$, taking the $(1-\delta)$ quantile z' in the empirical distribution and returning k/z'.

From simulations we obtain that for $\delta = 0.01$ and $\rho \in \{1, 2\}$, $C = 2$ suffices for sample size $k \geq 10$, $C = 1.4$ suffices for $k \geq 100$, and $C = 1.1$ suffices for $k \geq 1000$.

C Domination of the ratio distribution

Lemma C.1 (Domination)**.** *For any permutation π, $\boldsymbol{w}$, p, $q \geq p$, and $k \geq 1$, the distribution $F_{\boldsymbol{w},p,q,k}|\pi$ (18) is dominated by $R_{n,k,q/p}$. That is, for all $z \geq 0$,*

$$\Pr_{z\sim F_{\boldsymbol{w},p,q,k}|\pi}\left[z \leq \frac{k}{\psi}\right] \geq \Pr_{z\sim R_{n,k,q/p}}\left[z \leq \frac{k}{\psi}\right] \tag{22}$$

Proof. Assume without loss of generality that $\mathsf{order}(\boldsymbol{w}) = \pi$. Let $\boldsymbol{w}^* \sim p\text{-ppswor}[\boldsymbol{w}] \mid \mathsf{order}(\boldsymbol{w}^*) = \pi$. Note by definition $\boldsymbol{w}^*$ is in decreasing order of magnitude. Define the random variable $\boldsymbol{y} := \boldsymbol{w}^{*p}$. $\boldsymbol{y}$ are transformed weights of a ppswor sample of $\boldsymbol{w}$ conditioned on the order π. We use use properties of the exponential distribution (see a similar use in [26]) to express the joint distribution of $\{y_i\}$. We use the following set of independent random variables:

$$X_i \sim \mathsf{Exp}[\sum_{j=i}^n w_j^p] \ .$$

We have:

$$y_i = \frac{1}{\left(\sum_{j=1}^i X_i\right)^{q/p}} \ . \tag{23}$$

To see this, recall that y_1 is the (inverse) of the minimum of exponential random variables with parameters $w_1, \ldots, w_n$ and thus is (the inverse of) exponential random variable with parameter equal to their sum. Therefore, $y_1 = 1/X_1$. From memorylessness, the difference between the $(i+1)$-st smallest inverse and the i-th smallest is an exponential random variable with distribution X_i. Therefore, the i-th smallest inverse has the claimed distribution (23).

We are now ready to express the random variable that is the ratio (18) in terms of the independent random variables X_i:

$$\frac{\sum_{j=k+1}^n y_j}{y_k} = \frac{\sum_{i=k+1}^n \frac{1}{\left(\sum_{j=1}^i X_i\right)^{q/p}}}{\frac{1}{\left(\sum_{j=1}^k X_j\right)^{q/p}}} = \sum_{i=k+1}^n \frac{\left(\sum_{j=1}^k X_j\right)^{q/p}}{\left(\sum_{j=1}^i X_j\right)^{q/p}} \ . \tag{24}$$

We rewrite this using i.i.d. random variables $Z_i \sim \mathsf{Exp}[1]$, recalling that for any w, $\mathsf{Exp}[w]$ is the same as $\mathsf{Exp}[1]/w$. Then we have $X_i = Z_i / \sum_{j=i}^n w_j^p$.

We next provide a simpler distribution that dominates the distribution of the ratio. Let $W' := \sum_{j=k}^n w_j^p$ and consider the i.i.d. random variables $X_i' = Z_i/W'$. Note that $X_j \leq X_j'$ for $j \leq k$ and $X_j \geq X_j'$ for $j \geq k$. Thus, for $i \geq k+1$,

$$\frac{\sum_{j=1}^k X_j}{\sum_{j=1}^i X_j} = \frac{1}{1 + \frac{\sum_{j=k+1}^i X_j}{\sum_{j=1}^k X_j}} \geq \frac{1}{1 + \frac{\sum_{j=k+1}^i X_j'}{\sum_{j=1}^k X_j'}} = \frac{\sum_{j=1}^k X_j'}{\sum_{j=1}^i X_j'} = \frac{\sum_{j=1}^k Z_j}{\sum_{j=1}^i Z_j} \tag{25}$$

This holds in particular for each term in the RHS of (24). Therefore we obtain

$$\frac{\sum_{j=k+1}^{n} y_j}{y_k} \geq \sum_{i=k+1}^{n} \frac{\left(\sum_{j=1}^{k} Z_j\right)^{q/p}}{\left(\sum_{j=1}^{i} Z_j\right)^{q/p}} \ .$$

□

D Tail bounds on $R_{k,n,\rho}$

We establish the following upper tail bounds on the distribution $R_{n,k,\rho}$:

Theorem D.1 (Concentration of $R_{n,k,\rho}$)**.** *There is a constant C, such that for any n, k, ρ*

$$\rho = 1\text{:} \quad \Pr_{r\sim R_{n,k,\rho}}\left[r \geq Ck\ln(\frac{n}{k})\right] \leq 3e^{-k} \tag{26}$$

$$\rho > 1\text{:} \quad \Pr_{r\sim R_{n,k,\rho}}\left[r \geq Ck\frac{1}{\rho-1}\right] \leq 3e^{-k} \tag{27}$$

We start with a "back of the envelope" calculation to provide intuition: replace the random variables Z_i in $R_{n,k,\rho}$ (see Definition B.1) by their expectation $\mathsf{E}[Z_i] = 1$ to obtain

$$S_{n,k,\rho} := \sum_{i=k+1}^{n} \frac{k^\rho}{i^\rho} \ .$$

For $\rho = 1$, $S_{n,k,\rho} \leq k(H_n - H_k) \approx k\ln(n/k)$. For $\rho > 1$ we have $S_{n,k,\rho} \approx \frac{k}{\rho-1}$. We will see that we can expect the sums not to deviate too far from this value.

The sum of ℓ i.i.d. $\mathsf{Exp}[1]$ random variables generates an Erlang distribution $\mathsf{Erlang}[\ell, 1]$ (rate parameter 1). The expectation is $\mathsf{E}_{r\sim\mathsf{Erlang}[\ell,1]} = \ell$ and variance is $\mathsf{Var}_{r\sim\mathsf{Erlang}[\ell,1]}[r] = \ell$. We will use the following Erlang tail bounds [48]:

Lemma D.1. *For $X \sim$ Erlang$[\ell, 1]$*

$$\begin{aligned} \varepsilon \geq 1: &\quad \Pr[x \geq \varepsilon\ell] \leq \tfrac{1}{\varepsilon}e^{-\ell(\varepsilon-1-\ln\varepsilon)} \leq e^{1-\varepsilon} \\ \varepsilon \leq 1: &\quad \Pr[x \leq \varepsilon\ell] \leq e^{-\ell(\varepsilon-1-\ln\varepsilon)} \end{aligned}$$

When $\varepsilon < 0.159$ or $\varepsilon > 3.2$ we have the bound $e^{-\ell}$. For $\varepsilon > 3.2$ we also have $\frac{1}{\varepsilon}e^{-\ell(\varepsilon-2.2)}$

Proof of Theorem D.1. We bound the probability of a "bad event" which we define as the numerator being "too large" and denominators being too "small." More formally, the numerator is the sum $N = \sum_{i=1}^{k} Z_i$ and we define a bad event as $N \geq 3.2k$. Substituting $\varepsilon = 3.2$ and $\ell = k$ in the upper tail bounds from Lemma D.1, we have that the probability of this bad event is bounded by

$$\Pr_{r\sim\mathsf{Erlang}[k,1]}[r > k\varepsilon] \leq e^{-k} \ . \tag{28}$$

The denominators are prefix sums of of the sequence of random variables. We consider a partition the sequence $Z_{k+1}, \ldots, Z_n$ to consecutive stretches of size

$$\ell_h := 2^h k, (h \geq 1) \ .$$

We denote by S_h the sum of stretch h. Note that $S_h \sim \mathsf{Erlang}[\ell_h, 1]$ are independent random variables. We define a bad event as the probability that for some $h \geq 1$, $S_h \leq 0.15\ell_h = 0.14\ 2^h k$. From the lower tail bound of Lemma D.1, we have

$$\Pr[S_h \leq 0.15\ell_h] = \Pr_{r\sim\mathsf{Erlang}[\ell_h,1]}[r < 0.15\ell_h] \leq e^{-\ell_h} \leq e^{-2^h k} \ . \tag{29}$$

The combined probability of the union of these bad events (for the numerator and all stretches) is at most $e^{-k} + \sum_{h\geq 1} e^{-2^h k} \leq 3e^{-k}$.

We are now ready to compute probabilistic upper bound on the ratios when there is no bad event

$$\begin{aligned} R_{n,k,\rho} &\leq \sum_{h\geq 1} \ell_h \frac{N^\rho}{(N+\sum_{i<h} S_i)^\rho} \\ &\leq \sum_{h\geq 1} \ell_h \frac{(3.2k)^\rho}{\left(3.2k+0.15\sum_{i<h}\ell_i\right)^\rho} \\ &= 2k\frac{(3.2k)^\rho}{(3.2k)^\rho} + \sum_{h\geq 2} 2^h k \frac{(3.2k)^\rho}{(3.2k+(2^h-2)k)^\rho} \\ &\leq k(2+\sum_{h=2}^{\lceil \log_2(n/k)\rceil} 2^h \left(\frac{3.2}{2^h+1.2}\right)^\rho \leq k\left(2+3.2^\rho \sum_{h=2}^{\lceil \log_2(n/k)\rceil} 2^{-h(\rho-1)}\right) \end{aligned}$$

For $\rho = 1$ we have $O(k\log n)$. For $\rho > 1$, we have $O(k/(\rho-1))$. □

From the proof of Theorem D.1 we obtain:

Corollary D.2. *There is a constant B such that when there are no "bad events" in the sense of the proof of Theorem D.1*

$$\frac{\sum_{i=1}^k Z_i}{\sum_{i=1}^{Bk} Z_i} \leq 1/3\,.$$

Proof. With no bad events, $N = \sum_{i=1}^k Z_i < 3.2k$ and $\sum_{i=k+1}^{k2^h} Z_i \geq 0.15k(2^h-1)$. Solving for $0.15kB \geq 6.4k$ (for $B = 2^h - 1$ for some h) we obtain $B = 63$. □

E Ratio of magnitudes of transformed weights

For $k_2 > k_1$ we consider the distribution of the ratio between the k_2^{th} and k_1^{th} transformed weights:

$$G_{\boldsymbol{w},p,q,k_1,k_2} \mid \pi := \left|\frac{w^{*p}_{(k_2)}}{w^{*p}_{(k_1)}}\right|$$

Lemma E.1. *For any $\boldsymbol{w} \in \Re^n$, $\pi \in S^n$, and $k_1 < k_2 \leq n$, the distribution $G_{\boldsymbol{w},p,q,k_1,k_2} \mid \pi$ is dominated by*

$$G'_{\rho=q/p,k_1,k_2} := \left(\frac{\sum_{i=1}^{k_1} Z_i}{\sum_{i=1}^{k_2} Z_i}\right)^\rho \quad , \tag{30}$$

where $Z_i \sim \mathsf{Exp}[1]$ are i.i.d.

Proof. Following the notation in the proof of Lemma C.1, the distribution $G_{\boldsymbol{w},p,q,k_1,k_2}$ can be expressed as

$$\left(\frac{\sum_{i=1}^{k_1} X_i}{\sum_{i=1}^{k_2} X_i}\right)^\rho$$

where $X_i := \frac{Z_i}{\sum_{j=i}^n w_j^p}$.

For $i \in [n]$ we define $X'_i := \frac{Z_i}{\sum_{j=k_1}^n w_j^p}$. Now note that $X'_i \geq X_i$ for $i \leq k_1$ and $X'_i \geq X_i$ for $i \geq k_1$. Therefore,

$$\begin{aligned} \frac{\sum_{i=1}^{k_1} X_i}{\sum_{i=1}^{k_2} X_i} &= \frac{1}{1+\frac{\sum_{i=k_1+1}^{k_2} X_i}{\sum_{i=1}^{k_1} X_i}} \\ &\leq \frac{1}{1+\frac{\sum_{i=k_1+1}^{k_2} X'_i}{\sum_{i=1}^{k_1} X'_i}} = \frac{1}{1+\frac{\sum_{i=k_1+1}^{k_2} Z_i}{\sum_{i=1}^{k_1} Z_i}} = \frac{\sum_{i=1}^{k_1} Z_i}{\sum_{i=1}^{k_2} Z_i}\,. \end{aligned}$$

□

F 1-pass with total variation distance on sample k-tuple: upper and lower bounds

Perfect ppswor returns each subset of k keys $S = \{i_1, \ldots, i_k\}$ with a certain probability:

$$p(S) = \sum_{\pi | \{\pi_1, \ldots, \pi_k\} = S} \prod_{j=1}^{k} \frac{w_{i_j}}{\|\boldsymbol{w}\|_1 - \sum_{h<j} w_{i_h}} .$$

Recall that the distribution is equivalent to successive weighted sampling without replacement. It is also equivalent to successive sampling with replacement if we "skip" repetitions until we obtain k distinct keys.

With p-ppswor and unaggregated data, this is with respect to ν_x^p. The WORp 1-pass method returns an approximate p-ppswor sample in terms of estimation quality and per-key inclusion probability but the TV distance on k-tuples can be large.

We present here another 1-pass method that returns a k-tuple with a polynomially small VT distance from p-ppswor.

Algorithm 1: 1-pass Low Variation Distance Sampling

Input: ℓ_p rHH method, perfect ℓ_p-single sampler method, sample size k, p, δ, n,

Initialization:

Initialize $r = C \cdot k \log n$ independent perfect ℓ_p-single sampling algorithms $A^1, \ldots, A^r$.

Initialize an ℓ_p rHH method R.

Pass 1:

Feed each stream update into $A^1, \ldots, A^r$ as well as into R.

Produce sample:

$S \leftarrow \emptyset$

For $i = 1, \ldots, r$

Let Out_i be the index returned by A^i

If $Out_i \notin S$, then

$S \leftarrow S \cup \{Out_i\}$

For each $j > i$, feed the update $x_{Out_i} \leftarrow x_{Out_i} - R(Out_i)$ into A^j // $R(Out_i)$ is the estimate of x_i given by R

If $|S| = k$ then exit and return S

end

Output FAIL // Algorithms returns S before reaching this line with high probability

Theorem F.1. *Let $p \in (0, 2]$. There is a 1-pass turnstile streaming algorithm using $k \cdot poly(\log n)$ bits of memory which, given a stream of updates to an underlying vector $x \in \{-M, -M+1, \ldots, M-1, M\}^n$, with $M = poly(n)$, outputs a set S of size k such that the distribution of S has variation distance at most $\frac{1}{n^C}$ from the distribution of a sample without replacement of size k from the distribution $\mu = (\mu_1, \ldots, \mu_n)$, where $\mu_i = \frac{|x_i|^p}{\|x\|_p^p}$, where $C > 0$ is an arbitrarily large constant.*

Proof. The algorithm is 1-pass and works in a turnstile stream given an ℓ_p rHH method and perfect ℓ_p-single sampler methods that have this property. We use the ℓ_p rHH method of [52], which has this property and uses $O(k \cdot \log^2 n)$ bits of memory. We also use the perfect ℓ_p-single sampler method of [50], which has this property and uses $\log^2 n \cdot \text{poly}(\log \log n)$ bits of memory for $0 < p < 2$ and $O(\log^3 n)$ bits of memory for $p = 2$. The perfect ℓ_p-single sampler method of [50] can output FAIL with constant probability, but we can repeat it $C \log n$ times and output the first sample found, so that it outputs FAIL with probability at most $\frac{1}{n^C}$ for an arbitrarily large constant $C > 0$, and consequently we can assume it never outputs FAIL (by say, always outputting index 1 when FAIL occurs). This gives us the claimed $k \cdot \text{poly}(\log n)$ total bits of memory.

We next state properties of these subroutines. The ℓ_p rHH method we use satisfies: with probability $1 - \frac{1}{n^C}$ for an arbitrarily large constant $C > 0$, simultaneously for all $j \in [n]$, it outputs an estimate $R(j)$ for which

$$R(j) = x_i \pm \left(\frac{1}{2k}\right)^{1/p} \cdot \|\mathsf{tail}_k(x)\|_p.$$

We assume this event occurs and add $\frac{1}{n^C}$ to our overall variation distance.

The next property concerns the perfect ℓ_p-single samplers A^j we use. Each A^j returns an index $i \in \{1, 2, \ldots, n\}$ such that the distribution of i has variation distance at most $\frac{1}{n^C}$ from the distribution μ. Here $C > 0$ is an arbitrarily large constant of our choice.

We next analyze our procedure for producing a sample. Consider the joint distribution of $(Out_1, Out_2, \ldots, Out_{2Ck \log n})$. The algorithms A^i use independent randomness. However, the input to A^i may depend on the randomness of $A^{i'}$ for $i' < i$. However, by definition, conditioned on A^i not outputting $Out_{i'}$ for any $i' < i$, we have that Out_i is independent of $Out_1, \ldots, Out_{i-1}$ and moreover, the distribution of Out_i has variation distance $\frac{1}{n^C}$ from the distribution of a sample s from μ conditioned on $s \notin \{Out_1, \ldots, Out_{i-1}\}$, for an arbitrarily large constant $C > 0$.

Let E be the event that we sample k distinct indices, i.e., do not output FAIL in our overall algorithm. We show below that $\Pr[E] \geq 1 - \frac{1}{n^C}$ for an arbitrarily large constant $C > 0$. Consequently, our output distribution has variation distance $1n^C$ from an idealized algorithm that samples until it has k distinct values.

Consider the probability of outputting a particular ordered tuple $(i_1, \ldots, i_k)$ of k distinct indices in the idealized algorithm that samples until it has k distinct values. By the above, this is

$$\prod_{j=1}^{k} (1 \pm \frac{1}{n^C}) \frac{\mu_{i_j}}{1 - \sum_{j' < j} \mu_{i_{j'}}} = (1 \pm \frac{2k}{n^C}) \prod_{j=1}^{k} \frac{\mu_{i_j}}{1 - \sum_{j' < j} \mu_{i_{j'}}},$$

for an arbitrarily large constant $C > 0$. Summing up over all orderings, we obtain the probability of obtaining the sample $\{i_1, \ldots, i_k\}$ is within $(1 \pm \frac{1}{n^C})$ times its probability of being sampled from μ in a sample without replacement of size k, where $C > 0$ is a sufficiently large constant.

It remains to show $\Pr[E] \leq n^{-C}$ for an arbitrarily large constant $C > 0$. Here we use that for all $j \in \{1, 2, \ldots, n\}$, $R(j) = x_i \pm \left(\frac{1}{2k}\right)^{1/p} \cdot \|\mathsf{tail}_k(x)\|_p$. Let Y_i be the number of trials until (and including the time) we sample the i-th distinct item, given that we have just sampled $i - 1$ distinct items. The total probability mass on the items we have already sampled is at most $i \cdot \frac{1}{2k} \|\mathsf{tail}_k(x)\|_p^p$, and thus the probability we re-sample an item already sampled is at most $\frac{1}{2}$. It follows that $\mathbf{E}[Y_i] \leq 2$. Thus, the number of trials in the algorithm is stochastically dominated by $\sum_{i=1}^{k} Z_i$, where Z_i is a geometric random variable with $\mathbf{E}[Z_i] = 2$. This sum is a negative binomial random variable, and by standard tail bounds relating a sum of independent geometric random variables to binomial random variables[62] [3], is at most $Ck \log n$ with probability $1 - \frac{1}{n^C}$ for an arbitrarily large constant $C > 0$.

This completes the proof. □

We now analyze the memory in Theorem F.1 more precisely. Algorithm 1 runs $r = O(k \log n)$ independent perfect ℓ_p-sampling algorithms of [50]. The choice of $r = O(k \log n)$ is to ensure that the variation distance is at most $\frac{1}{\mathrm{poly}(n)}$; however, with only $r = O(k)$ such samplers, the same argument as in the proof of Theorem F.1 gives variation distance at most $2^{-\Theta(k)}$. Now, each ℓ_p-sampler of [50] uses $O(\log^2 n (\log \log n)^2)$ bits of memory for $0 < p < 2$, and uses $O(\log^3 n)$ bits of memory for $p = 2$. We also do not need to repeat the algorithm $O(\log n)$ times to create a high probability of not outputting FAIL; indeed, already if with only constant probability the algorithm does not output FAIL, we will still obtain k distinct samples with $2^{-\Theta(k)}$ failure probability provided we have a large enough $r = O(k)$ number of such samplers.

Algorithm 1 also runs an ℓ_p rHH method, and this uses $O(k \log^2 n)$ bits of memory [52]. Consequently, to acheive variation distance at most $2^{-\Theta(k)}$, Algorithm 1 uses $O(k \log^2 n (\log \log n)^2)$ bits of memory for $0 < p < 2$, and $O(k \log^3 n)$ bits of memory for $p = 2$.

We now show that for $0 < p < 2$, the memory used of Algorithm 1 is best possible for any algorithm, up to a multiplicative $O((\log \log n)^2)$ factor. For $p = 2$, we show our algorithm's memory is optimal up to a multiplicative $O(\log n)$ factor. Further, our lower bound holds even for any algorithm with the much weaker requirement of achieving variation distance at most $\frac{1}{3}$, as opposed to the variation distance at most $2^{-\Theta(k)}$ that we achieve.

Theorem F.2. *Any 1-pass turnstile streaming algorithm which outputs a set S of size k such that the distribution of S has variation distance at most $\frac{1}{3}$ from the distribution of a sample without replacement of size k from the distribution $\mu = (\mu_1, \ldots, \mu_n)$, where $\mu_i = \frac{|x_i|^p}{\|x\|_p^p}$, requires $\Omega(k \log^2 n)$ bits of memory, provided $k < n^{C_0}$ for a certain absolute constant $C_0 > 0$.*

[3] See, also, e.g., https://math.stackexchange.com/questions/1565559/tail-bound-for-sum-of-geometric-random-variables

Proof. We use the fact that such a sample S can be used to find a constant fraction of the $\ell_q(k,1)$ residual heavy hitters in a data stream. Here we do not need to achieve residual error for our lower bound, and can instead define such indices i to be those that satisfy $|x_i|^p \geq \frac{1}{k}\|x\|_p^p$. Notice that there are at most k such indices i, and any sample S (with or without replacement) with variation distance at most $1/3$ from a true sample has probability at least $1-(1-1/k)^k - 1/3 \geq 1 - 1/e - 1/3 \geq .29$ of containing the index i. By repeating our algorithm $O(1)$ times, we obtain a superset of size $O(k)$ which contains any particular such index i with arbitrarily large constant probability, and these $O(1)$ repetitions only increase our memory by a constant factor.

It is also well-known that there exists a point-query data structure, in particular the `CountSketch` data structure [15, 67], which only needs $O(\log |S|) = O(\log k)$ rows and thus $O((k \log k)\log n)$ bits of memory, such that given all the indices j in a set S, one can return all items $j \in S$ for which $|x_j|^p \geq \frac{1}{k}\|x\|_p^p$ and no items $j \in S$ for which $|x_j|^p < \frac{1}{2k}\|x\|_p^p$. Here we avoid the need for $O(\log n)$ rows since we only need to union bound over correct estimates in the set S.

In short, the above procedure allows us to, with arbitrarily large constant probability, return a set S containing a random .99 fraction of the indices j for which $|x_j|^p \geq \frac{1}{k}\|x\|_p^p$, and containing no index j for which $|x_j|^p < \frac{1}{2k}\|x\|_p^p$.

We now use an existing $\Omega(k \log^2 n)$ bit lower bound, which is stated for finding all the heavy hitters [52], to show an $\Omega(k \log^2 n)$ bit lower bound for the above task. This is in fact immediate from the proof of Theorem 9 of [52], which is a reduction from the one-way communication complexity of the Augmented Indexing Problem and just requires any particular heavy hitter index to be output with constant probability. In particular, our algorithm, combined with the $O((k \log k)\log n)$ bits of memory side data structure of [15] described above, achieves this.

Consequently, the memory required of any 1-pass streaming algorithm for the sampling problem is at least $\Omega(k \log^2 n) - O((k \log k)\log n)$ bits, which gives us an $\Omega(k \log^2 n)$ lower bound provided $k < n^{C_0}$ for an absolute constant $C_0 > 0$, completing the proof. □

G Estimates of one-pass WORp

We first review the setup. Our one-pass WORp method returns the top k keys by $\widehat{\nu_x^*}$ as our sample S and returns $\widehat{\nu^*}_{(k+1)}$ as the threshold. The estimate of $f(\nu_x)$ is 0 for $x \notin S$ and for $x \in S$ is

$$\widehat{f(\nu_x)} := \frac{f(\widehat{\nu_x^*} r_x^{1/p})}{1 - \exp\left(-r_x(\frac{\widehat{\nu_x^*}}{\widehat{\nu^*}_{(k'+1)}})^p\right)} \quad . \tag{31}$$

We assume that $f(w)$ is such that for some constant c,

$$\forall \varepsilon < 1/2, |f((1+\varepsilon)w) - f(w)| \leq c\varepsilon f(w) \quad . \tag{32}$$

We need to establish that

$$\begin{aligned} \mathsf{Bias}[\widehat{f(\nu_x)}] &\leq O(\varepsilon) f(\nu_x) \\ \mathsf{MSE}[\widehat{f(\nu_x)}] &\leq (1+O(\varepsilon))\mathsf{Var}[\widehat{f(\nu_x)}'] + O(\varepsilon) f(\nu_x)^2 \quad , \end{aligned}$$

where $\widehat{f(\nu_x)}'$ are estimates obtained with a (perfect) p-ppswor sample.

Proof of Theorem 5.1. From (10), the rHH sketch has the property that for all keys in the dataset,

$$\|\widehat{\boldsymbol{\nu}^*} - \boldsymbol{\nu}^*\|_\infty \leq \varepsilon \nu^*_{(k+1)} \quad . \tag{33}$$

For sampled keys, $|\nu_x^*| \geq |\nu^*_{(k+1)}|$ and hence $|\widehat{\nu_x^*} - \nu_x^*| \leq \varepsilon |\nu^*_{(k+1)}| \leq \varepsilon |\nu_x^*|$. Using (6) we obtain that $\|\nu'_x - \nu_x\| \leq \varepsilon |\nu_x|$. From our assumption (32) , we have $|f(\nu'_x) - f(\nu_x)| \leq c\varepsilon f(\nu_x)$.

We consider the inclusion probability and frequency estimate of a particular key x, conditioned on fixed randomization r_z of all other keys $z \neq x$. The key x is included in the sample if $\widehat{\nu_x^*} \geq \widehat{\boldsymbol{\nu}^*}_{(k+1)}$. We consider the distribution of $\widehat{\nu_x^*}$ as a function of $r_x \sim \mathsf{Exp}[1]$. The value has a form of $E + \nu_x / r_x^{1/p}$, where the erro E satisfies $|E| \leq \varepsilon |\nu^*_{(k+1)}|$. The

conditioned inclusion probability thus falls in the range

$$p'_x = \Pr[\nu_x/r_x^{1/p} \pm \varepsilon|\nu^*_{(k)}| \geq \widehat{\boldsymbol{\nu}^*}_{(k)}] = \Pr\left[r_x \leq \left(\frac{\nu_x}{\widehat{\boldsymbol{\nu}^*}_{(k+1)} \pm \varepsilon|\nu^*_{(k)}|}\right)^p\right] = 1 - \exp(-\left(\frac{\nu_x}{\widehat{\boldsymbol{\nu}^*}_{(k+1)} \pm \varepsilon|\nu^*_{(k)}|}\right)^p) \ .$$

We estimate p'_x by

$$p''_x = 1 - \exp\left(-r_x(\frac{\widehat{\nu^*_x}}{\widehat{\boldsymbol{\nu}^*}_{(k+1)}})^p\right) \ . \tag{34}$$

This estimate has a small relative error. This due to the relative error in $\widehat{\nu^*_x}$ and because $|(1-\exp(-(1\pm\epsilon)b))) - (1-\exp(-b))| = O(\epsilon)(1-\exp(-b))$ and $(\frac{\nu'_x}{\widehat{\boldsymbol{\nu}^*}_{(k)}})^p)$ is an $O(\epsilon)$ relative error approximation of $(\frac{\nu_x}{\widehat{\boldsymbol{\nu}^*}_{(k)}-E})^p$.

We first consider the bias. Instead of using the unbiased inverse probability estimate $f(\nu_x)/p'_x$ when x is sampled (with probability p'_x) our estimator (17) $(f(\nu'_x)/p''_x$ approximates both the numerator and the denominator.

In the numerator of the estimator, we replace $f(\nu_x)$ by the relative error approximation $f(\nu'_x)$. Therefore overall, we use a small relative error estimate of the actual inverse probability estimate when it is non zero, which translates to a bias that is $O(\epsilon)f(\nu_x)$.

We next bound the Mean Squared Error (MSE) of the estimator (17). We express the variance contribution of exact p-ppswor conditioned on the same randomization r_z of all keys $z \neq x$. This is $\widehat{\mathsf{Var}[f(\nu_x)}'] = (1/p_x - 1)f(\nu_x)^2$, where $p_x = \Pr[\nu_x/r_x^{1/p} \geq \boldsymbol{\nu}^*_{(k)}] = 1 - \exp(-\left(\frac{\nu_x}{\boldsymbol{\nu}^*_{(k)}}\right)^p)$. The MSE contribution is

$$p'_x(f(\nu'_x)/p''_x - f(\nu_x))^2 + (1-p'_x)f(\nu_x)^2 \ . \tag{35}$$

We observe that the approximate threshold (that determines p'_x) approximates the perfect p-ppswor threshold: $|\widehat{\boldsymbol{\nu}^*}_{(k)} - \boldsymbol{\nu}^*_{(k)}| \leq \varepsilon|\nu^*_{(k)}|$. When $p_x < 1/2$, (35) approximates $\widehat{\mathsf{Var}[f(\nu_x)}']$ with relative error $O(\varepsilon)$.

When p_x is close to 1 this is dominated by $O(\varepsilon)f(\nu_x)^2$.

□

We remark that our analysis of the error only assumed the rHH error bound (33) which holds for all sketch types including `Counters`. The bias analysis can be tightened for `CountSketch` that returns unbiased estimates of the frequency.

H Pseudocode

Algorithm 2: 2-pass WORp

Input: ℓ_q rHH method, sample size k, p, δ, n,
Initialization:
Draw a random hash $r_x \sim \mathcal{D}$ // Random map of keys x to r_x

$\psi \leftarrow \frac{1}{3}\Psi_{n,k,q/p}(\delta)$
Initialize KeyHash // random hash function from strings to $[n]$

R.Initialize(k, ψ) // Initialize rHH structure randomization
Pass I: // Use composable aggregation (process input keys into rHH structures and merge rHH structures)
begin
  Process data element $e = (e.\mathsf{key}, e.\mathsf{val})$ into rHH sketch R
  R.Process(KeyHash$(e.\mathsf{key}), e.\mathsf{val}/r_{e.\mathsf{key}}^{1/p}$) // Generate and process output element
Pass II: // For keys with top $2k$ estimates $\widehat{\nu_x^*}$, collect exact frequencies ν_x.

Initialize a composable top-$2k$ structure T. The structure stores for each key its priority and frequency. The structure collects exact frequencies for the keys with top $2k$ priorities. Merge(T_1, T_2): Add up values and retain $3k$ top priority keys.
Process data element $e = (e.\mathsf{key}, e.\mathsf{val})$ into T
begin
  if $e.key \in T$ **then**
    $T[e.\mathsf{key}].val += e.\mathsf{val}$
  else
    $est \leftarrow R$.Est[KeyHash$(e.\mathsf{key})$] // $\widehat{\nu_{\mathsf{key}}^*}$

    if $est >$ *lowest priority in* T **then**
      Insert $e.\mathsf{key}$ to T
      $T[e.\mathsf{key}].val \leftarrow e.\mathsf{val}$
      $T[e.\mathsf{key}].priority \leftarrow est$
      if $|T|>2k$ **then**
        Eject lowest priority key from T

Produce sample: Sort T by $T[x].val * r_x^{1/p}$ // actual ν_x^* for keys in T

Return $(x, T[x].val)$ for top-k keys and $(k+1)$th $T[x].val * r_x^{1/p}$