Stratified Random Sampling from Streaming and Stored Data

Trong Duc Nguyen
Iowa State University, USA

Ming-Hung Shih
Iowa State University, USA

Divesh Srivastava
AT&T Labs–Research, USA

Srikanta Tirthapura
Iowa State University, USA

Bojian Xu
Eastern Washington University, USA

ABSTRACT

Stratified random sampling (SRS) is a widely used sampling technique for approximate query processing. We consider SRS on continuously arriving data streams, and make the following contributions. We present a lower bound that shows that any streaming algorithm for SRS must have (in the worst case) a variance that is $\Omega(r)$ factor away from the optimal, where r is the number of strata. We present S-VOILA, a streaming algorithm for SRS that is *locally variance-optimal*. Results from experiments on real and synthetic data show that S-VOILA results in a variance that is typically close to an optimal offline algorithm, which was given the entire input beforehand. We also present a variance-optimal offline algorithm VOILA for stratified random sampling. VOILA is a strict generalization of the well-known *Neyman allocation*, which is optimal only under the assumption that each stratum is abundant, i.e. has a large number of data points to choose from. Experiments show that VOILA can have significantly smaller variance (1.4x to 50x) than Neyman allocation on real-world data.

1 INTRODUCTION

Random sampling is a widely-used method for data analysis, and features prominently in the toolbox of virtually every approximate query processing system. The power of random sampling lies in its generality. For many important classes of queries, an approximate answer whose error is small in a statistical sense can be efficiently obtained through executing the query over an appropriately derived random sample. Sampling operators are part of all major database products, *e.g.,* Oracle, Microsoft SQL Server, and IBM Db2. The simplest method for random sampling is uniform random sampling, where each element from the entire data (the "population") is chosen with the same probability. Uniform random sampling may however lead to a high variance in estimation. For instance, consider a population $D = \{1, 2, 4, 2, 1, 1050, 1000, 1200, 1300\}$, and suppose we wanted to estimate the population mean. A uniform random sample of size two leads to an estimate with a variance of approximately 1.6×10^5.

An alternative sampling method is *stratified random sampling* (SRS), where the population is partitioned into subgroups called "strata". From within each stratum, uniform random sampling is used to select a per-stratum sample. All per-stratum samples are combined to derive the "stratified random sample". Suppose that the population is divided into two strata, one with elements $\{1, 2, 4, 2, 1\}$ and the other with elements $\{1000, 1050, 1200, 1300\}$. A stratified random sample of size two that chooses one element from each stratum yields an estimate with variance 2.47×10^3, much smaller than a uniform random sample of the same size.

SRS provides the flexibility to emphasize some strata over others through controlling the allocation of sample sizes; for instance, a stratum with a high standard deviation can be given a larger allocation than another stratum with a smaller standard deviation. In the above example, if we desire a stratified sample of size three, it is best to allocate a smaller sample of size one to the first stratum and a larger sample size of two to the second stratum, since the standard deviation of the second stratum is higher. Doing so, the variance of estimate of the population mean further reduces to approximately 1.23×10^3. The strength of SRS is that a stratified random sample can be used to answer queries not only for aggregates such as the mean, and sum of the entire population, but also of subsets of the population defined by selection predicates that are provided at query time. SRS has been used widely in database systems for approximate query processing [1–3, 8, 14, 30, 31].

A problem with handling large streaming data is that current methods for SRS are predominantly offline methods that assume all data is available before sampling starts. As a result, systems that rely on SRS (*e.g.,* [3, 14, 31]) cannot easily adapt to new data and have to recompute stratified random samples from scratch, as more data arrives. With the advent of streaming data warehouses such as Tidalrace [29], it is imperative to have methods for SRS that work on dynamic data streams, and maintain stratified random samples in an incremental manner.

We address the shortcoming of current methods through a study of SRS on streaming data. The difficulty of SRS on streaming data is that there are two logical processes simultaneously at work. One is sample size allocation, which allocates samples among the different strata in a manner that minimizes the variance of an estimate. The second is the actual sampling of elements from within each stratum. While each of these two steps, sample size allocation and sampling, can be done individually in a streaming fashion, it is far more challenging to do them simultaneously. We present lower bounds as well as algorithms for the task of maintaining a stratified random sample on a data stream. The quality of a stratified random sample is measured using the variance of an estimate of a population statistic, computed using the sample.

1.1 Our Contributions

– **Streaming Lower Bound:** We present a lower bound showing that in the worst case, any streaming algorithm for SRS that uses a memory of M records must have a variance that is $\Omega(r)$ away from the variance of the optimal offline algorithm that uses the same memory of M records, where r is the number of strata. We show that this lower bound is tight, by construction.

– **Practical Streaming Algorithm for SRS:** We present S-VOILA (Streaming Variance OptImaL Allocation) a streaming algorithm for SRS that is locally variance-optimal. Upon receiving new elements, it (re-)allocates sample sizes among strata so as to obtain the smallest variance among all possible re-allocations. S-VOILA can also deal with the case when a minibatch of multiple data

Series ISSN: 2367-2005

10.5441/002/edbt.2019.04

items is seen at a time, as in systems such as Spark streaming [42]. Re-allocations made by S-VOILA are locally optimal with respect to the entire minibatch, and the quality of re-allocations improve as the minibatch size increases. Since S-VOILA can deal with minibatches of varying sizes, it is well-suited to real-world streams that may have bursty arrivals.

– **Variance-Optimal Sample Size Reduction:** The streaming algorithm (S-VOILA) re-allocates sample sizes based on a novel method for reducing the size of an existing stratified random sample down to a desired target size in a variance-optimal manner. This novel technique for sample size reduction may be of independent interest in other tasks, e.g., sub-sampling from a given stratified random sample.

– **Sampling from a Sliding Window of a Stream:** We present an algorithm for sampling from a sliding window of the most recent elements in a stream. This algorithm uses memory much smaller than the size of the window, and results in a sample whose variance is close to that obtained by an optimal offline algorithm that is allowed multiple passes through data in the window.

– **Variance Optimal Offline SRS:** We present the first offline algorithm for variance-optimal SRS. Our algorithm VOILA computes an allocation with provably optimal variance among all possible allocations of sample sizes to strata. The well known **Neyman Allocation** [36] (NeyAlloc), which originated from the statistics literature, assumes that each stratum has an abundance of data to choose from. However, this assumption may not hold in databases, since each stratum is a subset of a database table, and the size of a stratum may be small. VOILA does not make such assumptions, and computes a variance optimal allocation no matter how large/small the sizes of the strata. Hence, VOILA is a strict generalization of NeyAlloc. In addition, VOILA does not make any assumption on how data is stratified.

– **Experimental Evaluation:** We present a detailed experimental evaluation using real and synthetic data, considering both the quality of sample and accuracy of query answers using the sample. In our experimental study, we found that (a) the variance of S-VOILA is typically close to that of the optimal offline algorithm VOILA, and the allocation of S-VOILA also closely tracks that of VOILA. S-VOILA also improves significantly upon prior work [5]. The variance of S-VOILA improves as the size of the minibatch increases, and a minibatch of size 100 provides most of the benefits of S-VOILA. (b) Samples produced using S-VOILA yield accurate answers to a range of queries that involve a selection followed by aggregation, where the selection predicate is provided at query time, and the aggregation function can be one of sum, average, and standard deviation[1]. (c) In the offline setting, VOILA can have significantly smaller variance than NeyAlloc.

1.2 Related Work

Sampling has been widely used in approximate query processing on both static and streaming data [17, 28, 33, 38, 39]. The reservoir sampling [34, 41] algorithm for uniform sampling from a stream has been known for decades, and many variants and generalizations have been considered, such as weight-based sampling [11, 22], insertion and deletion of elements [25], distinct sampling [26], sampling from a sliding window [9, 12, 23], time-decayed sampling [19, 20], and distributed streaming sampling [15, 16, 18, 40].

SRS in the online setting can be viewed as weight-based reservoir sampling where the weight of each stream element depends on the stratum it belongs to. Since the weight of a stream element changes dynamically (even after it has been observed) prior work on weighted reservoir sampling [22] does not apply, since it assumes that the weight of an element is known at the time of observation and does not change henceforth. Meng [35] considered streaming SRS using population-based allocation. Al-Kateb *et al.* [4, 5] considered streaming SRS using power allocation, based on their prior work on adaptive reservoir sampling [6]. Lang *et al.* [32] consider machine learning methods for determining the per-item probability of inclusion in a sample. This work is meant for static data, and can be viewed as a version of weighted random sampling where the weights are learnt using a query workload. Prior work on streaming SRS neither considers provable guarantees on the quality of the resulting samples, nor lower bounds for streaming SRS, like we do here.

A majority of prior work on using SRS in approximate query processing [1–3, 8, 14, 30, 31] has assumed static data. With the emergence of data stream processing systems [7] and data stream warehousing systems [29], it is important to devise methods for streaming SRS with quality guarantees.

2 PRELIMINARIES

Stratified sampling can be viewed as being composed of three parts – stratification, sample allocation, and sampling. Stratification is a partitioning of the universe into a number of disjoint strata. Equivalently, it is the assignment of each data element to a unique stratum. In database applications, stratification is usually a pre-defined function of one or more attributes of the data [17]. For example, the works of Chaudhuri et al. [14] and Agarwal et al. [3] on approximate query answering stratify tuples in a database table based on the set of selection predicates in the query workload that the tuple satisfies, and the work of Kandula et al. [31] on approximate query answering stratify rows of a table using the group ids derived from a group-by query. Note that our methods do not assume that stratification is performed in any specific manner, and work regardless of the method used to stratify data.

Our work considers sample allocation, the partitioning of the available memory budget of M samples among the different strata. In streaming SRS, the allocation needs to be continuously re-adjusted as more data arrives, and the characteristics of different strata change. In offline sampling, allocation needs to be done only once, after knowing the data in its entirety.

The final sampling step considers each stratum and chooses the assigned number of samples uniformly at random. In offline stratified sampling, the sampling step can be performed in a second pass through the data using reservoir sampling on the subset of elements belonging to each stratum, after a first pass has determined the sample size allocation. In the case of streaming sampling, the sampling step needs to occur simultaneously with sample (re-)allocation, which may change allocations to different strata over time.

Variance-Optimal Allocation. The quality of a stratified random sample is measured through the variance of an estimate that is derived using the sample. Consider a data stream

[1] Note that a query for the variance or standard deviation of data is distinct from the variance or standard deviation of an estimate.

$R = \{v_1, v_2, \ldots, v_n\}$ of current size n, whose elements are stratified into r strata, numbered $1, 2, \ldots, r$. Let n_i denote the number of elements in stratum i. For each $i = 1 \ldots r$, let S_i be a uniform random sample of size s_i drawn without replacement from stratum i. Let $\mathbb{S} = \{S_1, S_2, \ldots, S_n\}$ denote the stratified random sample. The sample mean of each per-stratum sample S_i is: $\bar{y}_i = \frac{\sum_{v \in S_i} v}{s_i}$. The population mean of R, μ_R can be estimated as: $\bar{y} = \frac{\sum_{i=1}^{r} n_i \bar{y}_i}{n}$. It can be shown that the expectation of $\bar{y}$ equals μ_R. Given a memory budget of $M \leq n$ elements to store all the samples, so that $\sum_i s_i = M$, the following question of *variance-optimal allocation* of sample sizes has been considered in prior work [36]: *How to split the memory budget M among the s_is to minimize the variance of $\bar{y}$?* The variance of $\bar{y}$ can be computed as follows (e.g. see Theorem 5.3 in [17]):

$$V = V(\bar{y}) = \frac{1}{n^2} \sum_{i=1}^{r} n_i (n_i - s_i) \frac{\sigma_i^2}{s_i} = \frac{1}{n^2} \sum_{i=1}^{r} \frac{n_i^2 \sigma_i^2}{s_i} - \frac{1}{n^2} \sum_{i=1}^{r} n_i \sigma_i^2 \tag{1}$$

While the theory around SRS in both statistics and database communities has used the variance of the population mean as a minimization metric, variance-optimal SRS is useful for other types of queries as well, including predicate-based selection queries, sum queries across a subset of the strata, queries for the variance, and combinations of such queries [3, 14] – also see our experiments section.

NeyAlloc for Abundant Strata. Prior studies on variance-optimal allocation have primarily considered static data. Additionally, they assume that every stratum has a very large volume of data, so that there is no restriction on the size of a sample that can be chosen from this stratum. This may not be true for the scenario of databases. Especially in a streaming context, each stratum starts out with very little data. Given a collection of data elements R, we say that a stratum i is *abundant* if $n_i \geq M \cdot (n_i \sigma_i) / \left(\sum_{j=1}^{r} n_j \sigma_j \right)$. Otherwise, the stratum i is said to be *bounded*. Under the assumption that each stratum is abundant, the popularly used "Neyman Allocation" NeyAlloc [17, 36] minimizes the variance V, and allocates a sample size for stratum i as $M \cdot (n_i \sigma_i) / \left(\sum_{j=1}^{r} n_j \sigma_j \right)$. We note that NeyAlloc is no longer optimal if one or more strata are bounded. Our methods of sample size reduction and online (S-VOILA) and offline (VOILA) algorithms do not have this restriction and work under the general case whether or not strata are bounded.

Our solution to streaming SRS consists of two parts – sample size re-allocation, and per-stratum random sampling. Both parts execute continuously and in an interleaved manner. Sample size re-allocation is achieved using a reduction to a "sample size reduction" in a variance-optimal manner. Given a stratified random sample $\mathbb{S}_1$ of size larger than a target M, sample size reduction seeks to find a stratified sample $\mathbb{S}_2$ of size M that is a subset of $\mathbb{S}_1$ such that the variance of $\mathbb{S}_2$ is as small as possible.

Roadmap: In Section 3, we consider streaming SRS, and present a tight lower bound for any streaming algorithm, followed by S-VOILA an algorithm for streaming SRS. This uses as a subroutine a variance-optimal sample size reduction method that we describe in Section 4. We start with SingleElementSSR for reducing the size of the sample by one element, followed by a general algorithm SSR for reducing the size by $\beta \geq 1$ elements. We then present an algorithm MultiElementSSR with a faster runtime. We then consider the case of sliding windows in Section 5, followed by the optimal offline algorithm in Section 6. We present an experimental study of our algorithms in Section 7.

3 STREAMING SRS

We now consider SRS from a data stream, whose elements are arriving continuously. As more elements are seen, the allocations as well as samples need to be dynamically adjusted. We first note there is a simple two-pass streaming algorithm with optimal variance that uses $O(k + r)$ space, where k is the desired sample size and r the number of strata. In the first pass, the size, mean, and standard deviations of each stratum are computed using $O(r)$ space, constant space for each stratum. At the end of the first pass, the allocations to different strata are computed using an optimal offline algorithm, say VOILA. In the second pass, since the desired sample sizes are known for each stratum, samples are computed using reservoir sampling within the substream of elements belonging to each stratum. The above two-pass algorithm *cannot be* converted into a one-pass algorithm. The difficulty is that as more elements are seen, allocations to different strata may change, and the sampling rate within a stratum cannot in general be (immediately) dynamically adjusted in order to satisfy variance optimality. We first show a lower bound that it is in general not possible for any streaming algorithm to have optimal variance compared with an offline algorithm that is given the same memory.

3.1 A Lower Bound for Streaming SRS

Given a data stream $\mathcal{R}$ with elements belonging to r strata, and a memory budget of M elements, let V^* denote the optimal sample variance that can be achieved by an offline algorithm for SRS that may make multiple passes through data. Clearly, the sample produced by any streaming algorithm must have variance that is either V^* or greater. Suppose a stratified random sample R is computed by a streaming algorithm using memory of M elements. Let $V(R)$ denote the variance of this sample. For $\alpha \geq 1$, we say R *is an SRS with multiplicative error of α, if: (1) the sample within each stratum in R is chosen uniformly from all elements in the stratum, and (2) $V(R) \leq \alpha \cdot V^*$.*

Theorem 3.1. *Any streaming algorithm for maintaining an SRS over a stream with r strata using a memory of M elements must, in the worst case, result in a stratified random sample with a multiplicative error $\Omega(r)$.*

The idea in the proof is to construct an input stream with r strata where the variance of different strata are the same until a certain point in time, at which the variance of a single stratum starts increasing to a high value – a variance-optimal SRS will respond by increasing the allocation to this stratum. However, we show that a streaming algorithm is unable to do so quickly. Though a streaming algorithm may compute the variance-optimal allocation to different strata in an online manner, it cannot actually maintain these dynamically sized samples using limited memory.

Proof. Consider an input stream where for each $i = 1 \ldots r$, the ith stratum consists of elements in the range $[i, i + 1)$. The stream so far has the following elements. For each $i, 1 \leq i \leq r$, there are $(\alpha - 1)$ copies of element i and one copy of $(i + \varepsilon)$ where $\varepsilon = 1/(r - 1)$ and $\alpha \geq 3$. After observing these elements, for stratum i we have $n_i = \alpha$, $\mu_i = \left(i + \frac{\varepsilon}{\alpha} \right)$, and it can be verified that $\sigma_i = \frac{\sqrt{\alpha - 1}}{\alpha} \varepsilon$.

Since the total memory budget is M, at least one stratum (say, Stratum 1) has a sample size no more than M/r. Suppose an element of value $(2 - \varepsilon)$ arrives next. This element belongs to stratum 1. Let n'_1, μ'_1, and σ'_1 denote the new size, mean, and standard deviation of stratum 1 after this element arrives. We have $n'_1 = \alpha + 1$ and $\mu'_1 = 1 + \frac{1}{\alpha+1}$. It can be verified that $\sigma'_1 = \sqrt{\frac{\varepsilon^2+(1-\varepsilon)^2-\frac{1}{\alpha+1}}{\alpha+1}}$. It follows that:

$$(\alpha + 1)\sqrt{\frac{\frac{1}{2} - \frac{1}{\alpha+1}}{\alpha + 1}} \leq n'_1\sigma'_1 \leq (\alpha + 1)\sqrt{\frac{1 - \frac{1}{\alpha+1}}{\alpha + 1}} \qquad (2)$$

$$\implies \frac{\sqrt{\alpha}}{2} \leq n'_1\sigma'_1 \leq \sqrt{\alpha} \qquad \text{(Note: } \alpha > 2) \qquad (3)$$

In 2, the left inequality stands when $\varepsilon = 1/2$ and the right inequality stands when $\varepsilon = 0$ or 1. We also have: $\sum_{i=2}^{r} n_i\sigma_i = (r-1)\alpha\frac{\sqrt{\alpha-1}}{\alpha}\varepsilon = \sqrt{\alpha - 1}$, where we have used $\varepsilon = \frac{1}{r-1}$. Thus,

$$\frac{\sqrt{\alpha}}{2} \leq \sum_{i=2}^{r} n_i\sigma_i \leq \sqrt{\alpha} \qquad \text{(Note: } \alpha > 2) \qquad (4)$$

Let V denote the sample variance of $\mathcal{A}$ after observing the stream of $(r\alpha + 1)$ elements. Let V^* denote the smallest sample possible with a stratified random sample of size M on this data. Let $\Delta = \left(n'_1\sigma'^2_1 + \sum_{i=2}^{r} n_i\sigma_i^2\right)/n^2$.

We observe that after processing these $(r\alpha + 1)$ elements, the sample size $s_1 \leq M/r + 1$. Using this fact and the definition of sample variance in Eq 1:

$$V = \frac{1}{n^2}\left(\frac{n'^2_1\sigma'^2_1}{s_1} + \sum_{i=2}^{r}\frac{n_i^2\sigma_i^2}{s_i}\right) - \Delta \geq \frac{1}{n^2}\left(\frac{n'^2_1\sigma'^2_1}{\frac{M}{r}+1} + \sum_{i=2}^{r}\frac{n_i^2\sigma_i^2}{\frac{M}{r-1}}\right) - \Delta$$

$$\geq \frac{1}{n^2}\left(\frac{\alpha/4}{\frac{M}{r}+1} + \sum_{i=2}^{r}\frac{(\alpha-1)\varepsilon^2}{M/(r-1)}\right) - \Delta = \frac{1}{n^2}\left(\frac{\alpha/4}{\frac{M}{r}+1} + \frac{\alpha-1}{M}\right) - \Delta$$

On the other hand, the smallest sample variance V^* is achieved by using Neyman allocation. By Inequalities 3 and 4, we know that if Neyman allocation is for the current stream of $r\alpha + 1$ elements, stratum 1 uses at least $M/3$ memory space, whereas all other strata equally share at least $M/3$ elements since all $n_i\sigma_i$ are equal for $i = 2, 3, \ldots, r$. Using these observations into Equation 1:

$$V^* \leq \frac{1}{n^2}\left(\frac{n'^2_1\sigma'^2_1}{M/3} + \sum_{i=2}^{r}\frac{n_i^2\sigma_i^2}{M/(3(r-1))}\right) - \Delta$$

$$\leq \frac{1}{n^2}\left(\frac{\alpha}{M/3} + \sum_{i=2}^{r}\frac{(\alpha-1)\varepsilon^2}{M/(3(r-1))}\right) - \Delta = \left(\frac{1}{n^2}\frac{6\alpha-3}{M}\right) - \Delta$$

Since $\Delta \geq 0$ and $M > r$, we have: $\frac{V}{V^*} \geq \frac{V+\Delta}{V^*+\Delta} = \Omega(r)$. □

We note that the above lower bound is tight (up to constant factors). Consider the algorithm which always allocates M/r memory to each of r strata that have been observed so far. It can be verified that this algorithm has a variance within an $O(r)$ multiplicative factor of the optimal. While theoretically such an algorithm (which we call the "senate" algorithm due to allocating every stratum the same resources) meets the worst case lower bound, it performs poorly in practice, since it treats all strata equally, irrespective of their volume or variance (see the experiments section).

3.2 S-VOILA: Streaming Algorithm for SRS

We now present a streaming algorithm S-VOILA that can maintain a stratified random sample on a stream with a good (though not optimal) variance. Given a memory budget of M items, S-VOILA maintains a SRS of size M with the following properties: (1) the samples within each stratum are chosen uniformly from all the stream elements seen in the stratum so far, (2) the sizes of samples allocated to different strata adapt to new stream elements by making "locally optimal" decisions that lead to the best allocations given the new stream elements. S-VOILA conceptually has to solve two problems. One is sample size re-allocation among strata, and the second is uniform sampling within each stratum. Let $\mathcal{R}$ denote the stream observed so far, and $\mathcal{R}_i$ the elements in $\mathcal{R}$ that belong to stratum i.

We first consider sample size re-allocation. Suppose due to the addition of new elements, the stream went from $\mathcal{R}^1$ to $\mathcal{R}^2$, and suppose that the stratified random sample at $\mathcal{R}^1$ allocated sample sizes to strata in a specific manner, S^1. Due to the new elements, the sizes and variances of different strata change, and as a result, the optimal allocation of samples in $\mathcal{R}^2$ may be different from the previous allocation S^1. Our approach is to first add new elements to the sample, and then re-allocate sample sizes using a "variance-optimal sample size reduction" optimization framework. *Given a current allocation of sample sizes to different strata, suppose new elements are added to the sample, causing it to exceed a memory threshold M. What is a way to reduce the current sample to a sample of size M such that the variance of the new sample is as small as possible?* In the following section (Section 4), we present algorithms for sample size reduction.

The second issue is to maintain a uniform random sample S_i of $\mathcal{R}_i$ when s_i, the size of the sample is changing. A decrease in an allocation to s_i can be handled easily, through discarding elements from the current sample S_i until the desired sample size is reached. What if we need to increase the allocation to stratum i? If we simply start sampling new elements according to the higher allocation to S_i, then recent elements in the stream will be favored over the older ones, and the sample within stratum i is no longer uniformly chosen. In order to ensure that S_i is always chosen uniformly at random from $\mathcal{R}_i$, newly arriving elements in $\mathcal{R}_i$ need to be held to the same sampling threshold as older elements, even if the allotted sample size s_i increases. S-VOILA resolves this issue in the following manner. An arriving element from $\mathcal{R}_i$ is assigned a random "key" drawn uniformly from the interval $(0, 1)$. The sample is maintained using the following invariant: *S_i is the set of s_i elements with the smallest keys among all elements so far in $\mathcal{R}_i$*. It is easy to verify that this is indeed a uniform sample drawn without replacement from $\mathcal{R}_i$. The consequence of this strategy is that if we desire to increase the allocation to stratum i, it may not be accomplished immediately, since a newly arriving element in $\mathcal{R}_i$ may not be assigned a key that meets this sampling threshold. Instead, the algorithm has to wait until it receives an element in $\mathcal{R}_i$ whose assigned key is small enough. To ensure the above invariant, the algorithm maintains for each stratum i a variable d_i that tracks the smallest key of an element in $\mathcal{R}_i$ that is not currently included in S_i. If an arriving element in $\mathcal{R}_i$ has a key that is smaller than or equal to d_i, it is included within S_i; otherwise, it is not.

Algorithms 1 and 2 respectively describe the initialization and insertion of a minibatch of elements. S-VOILA supports the insertion of a minibatch of any size $b > 0$, where b can change from one minibatch to another. As b increases, we can expect

Algorithm 1: S-VOILA: Initialization

```
Input: $M$ – total sample size, $r$ – number of strata.
// $S_i$ is the sample for stratum $i$, and $\mathcal{R}_i$ is the
    substream of elements from Stratum $i$
Load the first $M$ stream elements in memory, and partition
  them into per-stratum samples, $S_1, S_2, \ldots, S_r$, such that $S_i$
  consists of $(e, d)$ tuples from stratum $i$, where $e$ is the
  element, $d$ is the key of the element, chosen independently
  and uniformly at random from $(0, 1)$.
For each stratum $i$, compute $n_i, \sigma_i$. Initialize $d_i \leftarrow 1$ ; // $d_i$
  tracks the smallest key among all elements in
  $\mathcal{R}_i$ not selected in $S_i$
```

Algorithm 2: S-VOILA: Process a new minibatch B of b elements. Note that b need not be known in advance, and can vary from one minibatch to the other.

```
1  $\beta \leftarrow 0$;                    // #selected elements from $B$
2  for each $e \in B$ do
3      Let $\alpha = \alpha(e)$ denote the stratum of $e$
4      Update $n_\alpha$ and $\sigma_\alpha$ ;  // per-stratum mean and std.
         dev. maintained in a streaming manner
5      Assign a random key $d \in (0, 1)$ to element $e$;
6      if $d \le d_\alpha$ then             // element $e$ is sampled
7          $S_\alpha \leftarrow \{e\} \bigcup S_\alpha$; $\beta \leftarrow \beta + 1$;
   /* Variance-optimal reduction by $\beta$ elements   */
8  if $\beta = 1$ then       // faster for evicting 1 element
9      $\ell \leftarrow$ SingleElementSSR($M$);
10     Delete one element of largest key from $S_\ell$;
11     $d_\ell \leftarrow$ smallest key discarded from $S_\ell$;
12 else if $\beta > 1$ then
13     $\mathcal{L} \leftarrow$ MultiElementSSR($M$);
14     for $i = 1 \ldots r$ do      // Actual element evictions
15         if $\mathcal{L}[i] < s_i$ then
16             Delete $s_i - \mathcal{L}[i]$ elements from $S_i$ with the
                 largest keys;
17             $d_i \leftarrow$ smallest key discarded from $S_i$;
```

S-VOILA to have a lower variance, since its decisions are based on greater amount of data. Lines 2–7 make one pass through the minibatch to update the mean and standard deviations of the strata, and store selected elements into the per-stratum samples. If $\beta > 0$ elements from the minibatch get selected into the sample, in order to balance the memory budget at M, β elements need to be evicted from the stratified random sample using the variance-optimal sample size reduction technique from Section 4.

A sample size reduction algorithm takes a current allocation to a stratified random sample, the statistics (volume, mean, and variance) of different strata, and a target sample size M, and returns the final allocation whose total size is M. For the special case of evicting one element, we can use the faster algorithm SingleElementSSR; otherwise, we can use MultiElementSSR. Lemma 3.2 shows that the sample maintained by S-VOILA within each stratum is a uniform random sample, showing this is a valid stratified sample, and Lemma 3.3 presents the time complexity analysis of S-VOILA. Proofs are omitted due to space constraints.

Lemma 3.2. *For each $i = 1, 2, \ldots, r$ sample S_i maintained by* S-VOILA *(Algorithm 2) is selected uniformly at random without replacement from stratum R_i.*

Lemma 3.3. *If the minibatch size $b = 1$, then the worst-case time cost of* S-VOILA *for processing an element is $O(r)$. The expected time for processing an element belonging to stratum α is $O(1 + r \cdot s_\alpha / n_\alpha)$, which is $O(1)$ when $r \cdot s_\alpha = O(n_\alpha)$. If $b > 1$, then the worst-case time cost of* S-VOILA *for processing a minibatch is $O(r \log r + b)$.*

We can expect S-VOILA to have an amortized per-item processing time of $O(1)$ in many circumstances. When $b = 1$: After observing enough stream elements from stratum α, such that $r \cdot s_\alpha = O(n_\alpha)$, the expected processing time of an element becomes $O(1)$. Even if certain strata have a very low frequency, the expected time cost for processing a single element is still expected to be $O(1)$, because elements from an infrequent stratum α are unlikely to appear in the minibatch. When $b > 1$: The per-element amortized time cost of S-VOILA is $O(1)$, when the minibatch size $b = \Omega(r \log r)$.

4 VARIANCE-OPTIMAL SAMPLE SIZE REDUCTION

Suppose it is necessary to reduce a stratified random sample (SRS) of total size M to an SRS of total size $M' < M$. This will need to reduce the size of the samples of one or more strata in the SRS. Since the sample sizes are reduced, the variance of the resulting estimate will increase. We consider the task of *variance-optimal sample size reduction (*VOR*), i.e.,* how to partition the reduction in sample size among the different strata in such a way that the increase in the variance is minimized. Note that once the new sample size for a given stratum is known, it is easy to subsample the stratum to the target sample size.

Consider Equation 1 for the variance of an estimate derived from the stratified random sample. Note that, for a given data set, a change in the sample sizes of different strata s_i does not affect the parameters n, n_i, and σ_i. VOR can be formulated as the solution to the following non-linear program.

$$\text{Minimize} \sum_{i=1}^{r} \frac{n_i^2 \sigma_i^2}{s_i'} \tag{5}$$

subject to constraints:

$$\sum_{i=1}^{r} s_i' = M' \qquad \text{and} \qquad 0 \le s_i' \le s_i \text{ for each } i = 1, 2, \ldots, r, \tag{6}$$

In the rest of this section, we present efficient approaches for computing the VOR.

4.1 Sample Size Reduction by One Element

We first present an efficient algorithm for the case where the size of a stratified random sample is reduced by one element. An example application of this case is in designing a streaming algorithm for SRS, when stream items arrive one at a time. The task is to choose a stratum i (and discard a random element from the stratum) such that after reducing the sample size s_i by one, the increase in variance V (Equation 1) is the smallest.

Our solution is to choose stratum i such that the partial derivative of V with respect to s_i is the largest over all possible choices of i.

$$\frac{\partial V}{\partial s_i} = -\frac{n_i^2 \sigma_i^2}{n^2} \frac{1}{s_i^2}.$$

Given a memory budget M and stratum i, let $M_i = M \cdot n_i\sigma_i/\sum_{j=1}^{r} n_j\sigma_j$ denote the amount of memory that NeyAlloc would allocate to stratum i. We choose stratum ℓ where:

$$\ell = \arg\max_i \left\{ \frac{\partial V}{\partial s_i} \,\middle|\, 1 \le i \le r \right\} = \arg\min_i \left\{ \frac{n_i\sigma_i}{s_i} \,\middle|\, 1 \le i \le r \right\}$$

Lemma 4.1. *When required to reduce the size of a stratified random sample by one, the increase in variance of the estimated population mean is minimized if we reduce the size of S_ℓ by one, where $\ell = \arg\min_i \left\{ \frac{n_i\sigma_i}{s_i} \,\middle|\, 1 \le i \le r \right\}$.*

In the case where we have multiple choices for ℓ using Lemma 4.1, we choose the one where the current sample size s_ℓ is the largest. Algorithm SingleElementSSR for reducing the sample by a single element is a direct implementation of the condition stated in Lemma 4.1. We omit the pseudocode due to space constraints. It is straightforward to observe this can be done in time $O(r)$.

4.2 Reduction by $\beta \ge 1$ Elements

We now consider the general case, where the sample needs to be reduced by $\beta \ge 1$ elements. A possible solution idea is to repeatedly apply the one-element reduction algorithm (Algorithm SingleElementSSRfrom Section 4.1) β times. Each iteration, a single element is chosen from a stratum such that the overall variance increases by the smallest amount. However, this greedy approach may not yield a sample with the smallest variance. On the other hand, an exhaustive search of all possible evictions is not feasible either, since the number of possible ways to partition a reduction of size β among r strata is $\binom{\beta+r-1}{r-1}$, which can be very large. For instance, if $r = 10$, this is $\Theta(\beta^{10})$. We now present efficient approaches to VOR. We first present a recursive algorithm, followed by a faster iterative algorithm. Before presenting the algorithm, we present the following useful characterization of a variance-optimal reduction.

Definition 4.2. We say that stratum i is *oversized* under memory budget M, if its allocated sample size $s_i > M_i$. Otherwise, we say that stratum i is *not oversized.*

Lemma 4.3. *Suppose that E is the set of β elements that are to be evicted from a stratified random sample such that the variance V after eviction is the smallest possible. Then, each element in E must be from a stratum whose current sample size is oversized under the new memory budget $M' = M - \beta$.*

Proof. We use proof by contradiction. Suppose one of the evicted elements is deleted from a sample S_α such that the sample size s_α is not oversized under the new memory budget. Because the order of the eviction of the β elements does not impact the final variance, suppose that element e is evicted after the other $\beta - 1$ evictions have happened. Let s_α denote the size of sample S_α at the moment t right after the first $\beta - 1$ evictions and before evicting e. The increase in variance caused by evicting an element from S_α is

$$\begin{aligned} \Delta &= \frac{1}{n^2}\left(\frac{n_\alpha^2\sigma_\alpha^2}{s_\alpha(s_\alpha - 1)}\right) = \left(\frac{\sum_{i=1}^{r} n_i\sigma_i}{nM'}\right)^2 \frac{M'^2_\alpha}{s_\alpha(s_\alpha - 1)} \\ &> \left(\frac{\sum_{i=1}^{r} n_i\sigma_i}{nM'}\right)^2 \end{aligned}$$

where ${M'}_\alpha = M' \frac{n_\alpha\sigma_\alpha}{\sum_{i=1}^{r} n_i\sigma_i}$. The last inequality is due to the fact that S_α is not oversized under budget M' at time t, i.e., $s_\alpha \le M'_\alpha$.

Note that an oversized sample exists at time t, since there are a total of $M' + 1$ elements in the stratified random sample at time t, and the memory target is M'. Instead of evicting e, if we choose to evict another element e' from an oversized sample $S_{\alpha'}$, the resulting increase in variance will be:

$$\begin{aligned} \Delta' &= \frac{1}{n^2}\left(\frac{n_{\alpha'}^2\sigma_{\alpha'}^2}{s_{\alpha'}(s_{\alpha'} - 1)}\right) = \left(\frac{\sum_{i=1}^{r} n_i\sigma_i}{nM'}\right)^2 \frac{M'^2_{\alpha'}}{s_{\alpha'}(s_{\alpha'} - 1)} \\ &< \left(\frac{\sum_{i=1}^{r} n_i\sigma_i}{nM'}\right)^2 \end{aligned}$$

where $M'_{\alpha'} = M' \frac{n_{\alpha'}\sigma_{\alpha'}}{\sum_{i=1}^{r} n_i\sigma_i}$ The last inequality is due to the fact that $S_{\alpha'}$ is oversized under budget M' at time t, i.e., $s_{\alpha'} > M'_{\alpha'}$. Because $\Delta' < \Delta$, at time t, evicting e' from $S_{\alpha'}$ leads to a lower variance than evicting e from S_α. This is a contradiction to the assumption that evicting e leads to the smallest variance, and completes the proof. □

Algorithm 3: SSR($\mathcal{A}, M, \mathcal{L}$): Variance-Optimal Sample Size Reduction

```
Input: $\mathcal{A}$ – set of strata under consideration.
       $M$ – target sample size for all strata in $\mathcal{A}$.
Output: For $i \in \mathcal{A}$, $\mathcal{L}[i]$ is the final size of sample for
        stratum $i$.
$O \leftarrow \emptyset$ // oversized samples
for $j \in \mathcal{A}$ do
    $M_j \leftarrow M \cdot n_j\sigma_j/\sum_{t\in\mathcal{A}} n_t\sigma_t$ // Neyman allocation
        if memory $M$ divided among $\mathcal{A}$
    if ($s_j > M_j$) then $O \leftarrow O \cup \{j\}$
    else $\mathcal{L}[j] \leftarrow s_j$ // Keep current allocation

if $O = \mathcal{A}$ then
    // All samples oversized. Recursion stops.
    for $j \in \mathcal{A}$ do $\mathcal{L}[j] \leftarrow M_j$
else
    // Recurse on $O$, under remaining mem budget.
    SSR($O, M - \sum_{j\in\mathcal{A}-O} s_j, \mathcal{L}$)
```

Lemma 4.3 implies that it is only necessary to reduce samples that are oversized under the target memory budget M'. Samples that are not oversized can be given their current allocation, even under the new memory target M'. Our algorithm based on this observation first allocates sizes to the samples that are not oversized. The remaining memory now needs to be allocated among the oversized samples. We note that this can again be viewed as a sample size reduction problem, while focusing on a smaller set of (oversized) samples, and accomplish it using a recursive call under a reduced memory budget; See Lemma 4.4 for a formal statement of this idea. The base case for this recursion is when all samples under consideration are oversized, in which case we can simply use NeyAlloc under the reduced memory budget M' (Observation 1). Our algorithm SSR is shown in Algorithm 3.

Let $\mathbb{S} = \{S_1, S_2, \ldots, S_r\}$ be the current stratified random sample. Let $\mathcal{A}$ denote the set of all strata under consideration, initialized to $\{1, 2, \ldots, r\}$. Let O denote the set of oversized samples, under target memory budget for $\mathbb{S}$, and $\mathcal{U} = \mathbb{S} - O$ denote the collection of samples that are not oversized. When the context is clear, we use O, $\mathcal{U}$, and $\mathcal{A}$ to refer to the set of stratum identifiers as well as the set of samples corresponding to these identifiers.

Table 1: An example of variance-optimal sample size reduction from 400×10^6 down to 200×10^6.

i	1	2	3	4	5	6
$n_i\sigma_i$ $(\times 10^9)$	10	8	30	20	8	24
s_i $(\times 10^6)$	15	50	50	45	60	180
round 1 M_i $(\times 10^6)$	20	< 50	60	< 45	< 60	< 180
round 2 M_i $(\times 10^6)$	-	< 50	-	45	< 60	< 180
round 3 M_i $(\times 10^6)$	-	18	-	-	18	54
s_i' $(\times 10^6)$	15	18	50	45	18	54

Lemma 4.4. *A variance-optimal eviction of β elements from $\mathbb{S}$ under memory budget M' requires a variance-optimal eviction of β elements from O under memory budget $M' - \sum_{j \in \mathcal{U}} s_j$.*

Proof. Recall that s_i' denotes the final size of sample S_i after β elements are evicted. Referring to the variance V from Equation 1, we know a variance-optimal sample size reduction of β elements from $\mathbb{S}$ under memory budget M' requires minimization of

$$\sum_{i\in\mathcal{A}} \frac{n_i^2\sigma_i^2}{s_i'} - \sum_{\in\mathcal{A}} \frac{n_i^2\sigma_i^2}{s_i} \qquad (7)$$

By Lemma 4.3, we know $s_i = s_i'$ for all $i \in \mathcal{U}$. Hence, minimizing Formula 7 is equivalent to minimizing

$$\sum_{i=O} \frac{n_i^2\sigma_i^2}{s_i'} - \sum_{i\in O} \frac{n_i^2\sigma_i^2}{s_i} \qquad (8)$$

The minimization of Formula 8 is exactly the result obtained from a variance-optimal sample size reduction of β elements from oversized samples under the new memory budget $M' - \sum_{i \in \mathcal{U}} s_i$. □

Observation 1. *In the case every sample in the stratified random sample is oversized under target memory M',* i.e., *$\mathbb{S} = O$, the variance-optimal reduction is to reduce the size of each sample $S_i \in \mathbb{S}$ to M_i' under the new memory budget M'.*

The following theorem summarizes the correctness and time complexity of Algorithm SSR.

Theorem 4.5. *Algorithm 3 (*SSR*) finds a variance-optimal reduction of the stratified random sample $\mathcal{A}$ under new memory budget M. The worst-case time of* SSR *is $O(r^2)$, where r is the number of strata.*

Proof. Correctness follows from Lemmas 4.3–4.4 and Observation 1. The worst-case time happens when each recursive call sees only one stratum that is not oversized. In such a case, the time of all recursions of SSR on a stratified random sample across r strata is: $O(r + (r-1) + \ldots + 1) = O(r^2)$. □

An Example (Table 1). Suppose we have 6 strata with their statistics ($n_i\sigma_i$) and current sample sizes (s_i) showin in Table 1 using a total size of $\sum_{i=1}^{6} s_i = 400$. Suppose that we wish to reduce the sample size down to 200 by reducing each s_i to the target sample size s_i'. The computation involves a sequence of recursive rounds. In the initial round, we allocate 200 samples among all 6 strata using Neyman allocation. Strata 1 and 3 turn out to be not oversized ($M_1 \geq s_1$, $M_3 \geq s_3$), and therefore we set $s_1' = s_1$ and $s_3' = s_3$. In Round 2, we exclude strata 1 and 3 from consideration, and the available memory budget which now becomes $200 - 15 - 50 = 135$. This is allocated among strata 2, 4, 5, and 6 using Neyman allocation. Stratum 4 is not oversized ($M_4 \geq s_4$) and therefore we set $s_4' = s_4$. At the next round 3, we further exclude stratum 4 from consideration, and the available memory budget now becomes $135 - 45 = 90$. When this is allocated among the remaining strata, it turns out that all of them are oversized ($M_i < s_i$, $i = 2, 5, 6$). We simply set $s_i' = M_i$ for each $i \in \{2, 5, 6\}$, and the recursion exits. Each stratum i now has a new sample size s_i' such that $s_i' \leq s_i$ for every i, and $\sum_{i=1}^{6} s_i' = 200$.

Algorithm 4: MultiElementSSR($\mathcal{A}$, M): A fast implementation of Sample Size Reduction without using recursion.

```
Input: The strata under consideration is $\mathcal{A} = \{1, 2, \ldots, r\}$,
       and the volumes and standard deviations. $M$ is the
       target total sample size.
Output: For $1 \leq i \leq r$, $\mathcal{L}[i]$ is set to the final size of sample
        for stratum $i$, such that the increase of the variance
        $V$ is minimized.
Allocate $\mathcal{L}[1..r]$, an array of numbers
Allocate $Q[1..r]$, an array of $(x, y, z)$ tuples
for $i = 1 \ldots r$ do $Q[i] \leftarrow (i, n_i\sigma_i, s_i/(n_i\sigma_i))$;
Sort array $Q$ in ascending order on the $z$ dimension
for $i = (r-1)$ down to 1 do
    $Q[i].y \leftarrow Q[i].y + Q[i+1].y$
$M_{new} \leftarrow M$; $D \leftarrow Q[1].y$
for $i = 1 \ldots r$ do
    $M_{Q[i].x} \leftarrow M \cdot n_{Q[i].x}\sigma_{Q[i].x}/D$
    if $s_{Q[i].x} > M_{Q[i].x}$ then break
    $\mathcal{L}[Q[i].x] \leftarrow s_{Q[i].x}$
    $M_{new} \leftarrow M_{new} - s_{Q[i].x}$
    // Check the next sample, which must exist.
    $M_{Q[i+1].x} \leftarrow M \cdot n_{Q[i+1].x}\sigma_{Q[i+1].x}/D$
    if $s_{Q[i+1].x} > M_{Q[i+1].x}$ then // oversized
        $M \leftarrow M_{new}$; $D \leftarrow Q[i+1].y$
// Reduce sample size to target.
for $j = i..r$ do
    // Desired size for $S_{Q[j].x}$
    $\mathcal{L}[Q[j].x] \leftarrow M \cdot n_{Q[j].x}\sigma_{Q[j].x}/D$
return $\mathcal{L}$
```

Faster Sample Size Reduction. We present a faster algorithm for variance-optimal sample size reduction, MultiElementSSR, with time complexity $O(r \log r)$. MultiElementSSR shares the same algorithmic foundation as SSR, but uses a faster iterative method based on sorting. We omit proofs due to space constraints.

Theorem 4.6. *(1) The* MultiElementSSR *procedure in Algorithm 4 finds the correct size of each sample of a stratified random sample, whose memory budget is reduced to M, such that the increase of the variance V is minimized. (2) The worst-case time cost of*MultiElementSSR *on a stratified random sample across r strata is $O(r \log r)$.*

5 STREAMING SRS OVER A SLIDING WINDOW

We consider the maintenance of an SRS drawn from a *sequence-based* sliding window of the most recent elements from the stream. Given a window size W, the sliding window consists of the W

most recent elements observed in the data stream. We consider the case when the window size W is very large, so that it is not feasible to store the entire window in memory. Similar to the algorithm for infinite window, there are two parts to the algorithm, sample re-allocation and sampling, which are interleaved with each other. We provide the algorithm idea and omit detailed descriptions.

For re-allocating sample sizes, we need the current statistics of each stratum within the sliding window. The mean and variance of a given stratum in an infinite window can be maintained in $O(1)$ space easily in a single pass. However, maintaining the mean and variance over a sliding window is much harder. In fact, it is known that exact computation of the mean and variance over a sliding window requires memory linear in the stream size [21] – thus, if we require these statistics exactly, we have to store the entire window, just to maintain the statistics of different strata! Fortunately, it is possible to approximate these statistics using space poly-logarithmic in the size of the stream; for the mean, see [21, 27], and for the variance [43].

Random sampling over a sliding window is also quite different from the case of infinite windows, and there is significant prior work on this e.g. [9, 13, 24]. We adapt algorithms from prior work to assign to each arriving element a random key, chosen uniformly in $[0, 1]$. The random sample of a certain size within a stratum is defined to be those elements in the stratum that have the smallest keys. Borrowing from prior work [9], we maintain additional recent elements within the window even if they don't belong to the set of keys that are currently the smallest – the reason is that these elements may become the elements with the smallest keys once the window slides and other elements with smaller keys "expire" from the window. The additional space required by these keys is a logarithmic factor in the size of the window (Section 2 in [9]). For each stratum, the algorithm continuously monitors the smallest key that has been discarded from the window.

When new elements arrive in the stream, these are sampled into the SRS, which may cause the size of the sample to increase beyond the memory allocated to the stratum. When this happens, we rely on variance-optimal sample size reduction (Algorithm `MultiElementSSR`) to give us new sample size allocations to different strata, and different strata are sub-sampled according to the new allocations (sub-sampling within a given stratum is handled through selecting only the elements with the smallest keys that are active in the window).

6 `VOILA`: VARIANCE-OPTIMAL OFFLINE SRS

We now present an algorithm for computing the variance-optimal allocation of sample sizes in the general case when there may be strata that are bounded. Note that once the allocation of sample sizes is determined, the actual sampling step is straightforward for the offline algorithm – samples can be chosen in a second pass through the data, using reservoir sampling within each stratum. Hence, in the rest of this section, we focus on determining the variance-optimal allocation. Consider a static data set R of n elements across r strata, where stratum i has n_i elements, and has standard deviation σ_i. *How can a memory budget of M elements be partitioned among the strata in a variance-optimal manner?* We present `VOILA` (**V**ariance-**O**pt**I**ma**L** **A**llocation), an efficient offline algorithm for variance-optimal allocation that can handle strata that are bounded.

Algorithm 5: `VOILA` (M): Variance-optimal stratified random sampling for bounded data

```
Input: $M$ is the memory target
for $i = 1 \ldots r$ do
    $s_i \leftarrow n_i$ // assume total memory of $n$
$\mathcal{L} \leftarrow$ MultiElementSSR($M$)
return $\mathcal{L}$    /* $\mathcal{L}[i] \leq n_i$ is the sample size for
    stratum $i$ in a variance-optimal stratified
    random sample. */
```

Neyman Allocation assumes there are no bounded strata (strata with small volumes). Note that it is not possible to simply eliminate strata with a low volume, by giving them full allocation, and then apply Neyman allocation on the remaining strata. The reason is as follows: suppose bounded strata are removed from further consideration. Then, remaining memory is divided among the remaining strata. This may lead to further bounded strata (which may not have been bounded earlier), and Neyman allocation again does not apply.

The following two-step process reduces variance-optimal offline SRS to variance-optimal sample size reduction.

Step 1: Suppose we start with a memory budget of n, sufficient to store all data. Then, we will just save the whole data set in the stratified random sample, and thus each sample size $s_i = n_i$. By doing so, the variance V is minimized, since $V = 0$ (Equation 1).

Step 2: Given the stratified random sample from Step 1, we reduce the memory budget from n to M such that the resulting variance is the smallest. This can be done using variance-optimal sample size reduction, by calling `SSR` or `MultiElementSSR` with target sample size M.

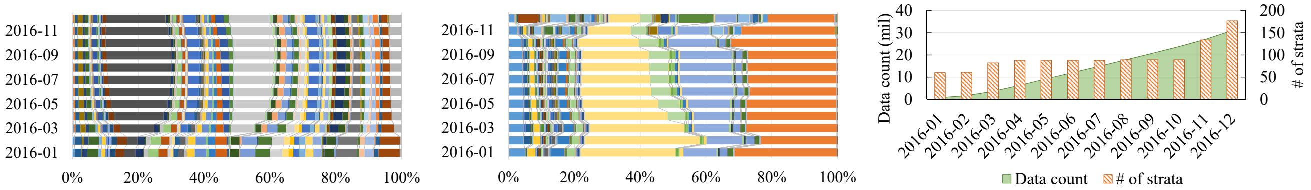
`VOILA` (Algorithm 5) simulates this process. The algorithm only records the sample sizes of the strata in array $\mathcal{L}$, without creating the actual samples. The actual sample from stratum i is created by choosing $\mathcal{L}[i]$ elements from stratum i, using a method for uniform random sampling without replacement.

Theorem 6.1. *Given a data set $\mathcal{R}$ with r strata, and a memory budget M, `VOILA` (Algorithm 5) returns in $\mathcal{L}$ the sample size of each stratum in a variance-optimal stratified random sample. The worst-case time cost of `VOILA` is $O(r \log r)$.*

Proof. The correctness follows from the correctness of Theorem 4.6, since the final sample is the sample of the smallest variance that one could obtain by reducing the initial sample (with zero variance) down to a target memory of size M. The run time is dominated by the call to `MultiElementSSR`, whose time complexity is $O(r \log r)$. □

7 EXPERIMENTAL EVALUATION

We present the results of an experimental evaluation. The input for our experiment is a (finite) stream of records from a data source, which is either processed by a streaming algorithm or by an offline algorithm at the end of computation. A streaming sampler must process data in a single pass using limited memory. An offline sampler has access to all data received, and can compute a stratified random sample using multiple passes through data. We evaluate the samplers in two ways. The first is a direct evaluation of the sample quality through the resulting allocation and the variance of estimates obtained using the samples. The second is through the accuracy of approximate query processing using the maintained samples for different queries.



(a) Relative (cumulative) frequencies of different strata. The x-axis is the fraction of points observed so far.

(b) Relative (cumulative) standard deviations of different strata.

(c) The number of strata received so far, and the number of records in data.

Figure 1: Characteristics of the OpenAQ dataset.

7.1 Sampling Methods

We compared our stream sampling method S-VOILA to Reservoir, ASRS and Senate sampling. Reservoir is a well-known stream sampling method that maintains a uniform random sample chosen without replacement from the stream - we expect the number of samples allocated to stratum *i* by Reservoir to be proportional to n_i. Senate [1] is a stratified sampling method that allocates each stratum an equal amount of sample space. For each stratum, Reservoir sampling is used to maintain a uniform sample.

ASRS is an adaptive stratified sampling algorithm due to Al-kateb *et al.* (Algorithm 3 in [5]). Their algorithm considers reallocations of memory among strata using a different method, based on power allocation [10], followed by reservoir sampling within each stratum. We chose the power allocation parameter to be 1 in order to obtain a sample of the entire population.

We also implemented three offline samplers VOILA, NeyAlloc, and an offline version of Senate. Each uses two passes to compute a stratified random sample of a total size of M records. The first pass is to determine strata characteristics used to allocate the space between strata. The second pass is to collect the samples accordingly to the computed allocation.

7.2 Data

We used a real-world dataset called OpenAQ [37], which contains more than 31 million records of air quality measurements (concentrations of different gases and particulate matter) from 7,923 locations in 62 countries around the world in 2016. Data is replayed in time order to generate the stream and is stratified based on the country of origin and the type of measurement, e.g., all measurements of carbon monoxide in the USA belong to one stratum, all records of sulphur dioxide in India belong to another stratum, and so on. The total number of strata at different points in time are shown in Figure 1c. We also experimented with another method of stratifying data, based only on the city of origin, whose results are shown at the end of this section. We also experimented with a synthetic dataset. The results obtained were qualitatively similar to the real-world data, and we omit these results due to space constraints.

Each stratum begins with zero records, and in the initial stages, every stratum is bounded. As more data are observed, many of the strata are not bounded anymore. As Figure 1c shows, new strata are added as more sensors are incorporated into the data stream. Figures 1a and 1b respectively show the cumulative frequency and standard deviation of the data over time; clearly these change significantly with time. As a result, the variance-optimal sample-size allocations to strata also change over time, and a streaming algorithm needs to adapt to these changes.

7.3 Allocations of Samples to Strata

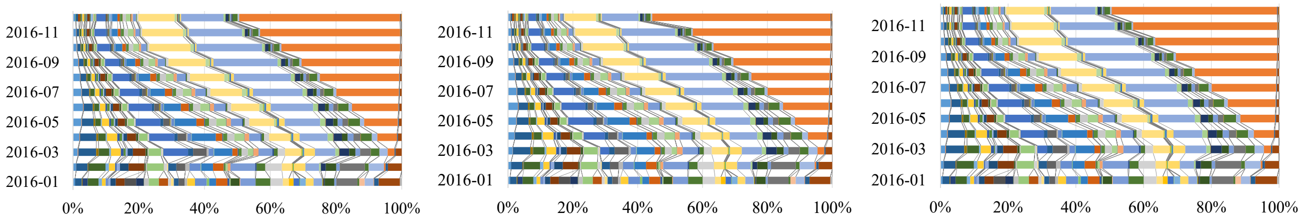
We measured the allocation of samples to different strata. Unless otherwise specified, the sample size M is set to 1 million records. For all experiments on allocations or variance, each data point is the mean of five independent runs. The allocation can be seen as a vector of numbers that sum up to M (or equivalently, normalized to sum up to 1), and we observe how this vector changes as more elements arrive.

Figures 2a, 2b and 2c show the change in allocations over time resulting from VOILA, S-VOILA with single element processing, and S-VOILA with minibatch processing. Unless otherwise specified, in the following discussion, the size of a minibatch is set to equal one day's worth of data. Visually, the allocations produced by the three methods track each other over time, showing that the streaming methods follow the allocation of VOILA. To understand the difference between the allocations due to VOILA and S-VOILA quantitatively, we measured the cosine distance between the allocation vectors from VOILA and S-VOILA. While detailed results are omitted due to space constraints, our results show that allocation vectors due to S-VOILA and VOILA are very similar, and the cosine distance is close to 0 most of the time and less than 0.04 at all times.

7.4 Comparison of Variance

We compared the variance of the estimates (Equation 1) produced by different methods. The results are shown in Figures 3 and 4. Generally, the variance of the sample due to each method increases over time, since the volume of data and the number of strata increase, while the sample size is fixed.

The comparison of different streaming algorithms is shown in Figure 4. Among the streaming algorithms, we first note that both variants of S-VOILA have a variance that is lower than ASRS, and typically close to the optimal (VOILA). The variance of S-VOILA with minibatch processing is typically better than with single element processing. We note that the variances of both variants of S-VOILA are nearly equal to that of VOILA until March, when they start increasing relative to VOILA, and then converge back. From analyzing the underlying data, we see that March is the time when a number of new strata appear in the data (Figure 1c), causing a substantial change in the optimal allocation of samples to strata. An offline algorithm such as VOILA can resample more elements at will, since it has access to all earlier data from the stratum. However, a streaming algorithm such as S-VOILA cannot do so and must wait for enough new elements to arrive in these strata before it can "catch up" to the allocation of VOILA. Hence, S-VOILA with single element as well as with minibatch processing show an increasing trend in the variance at such a point. When data becomes stable again the relative



(a) VOILA (b) S-VOILA with single element (c) S-VOILA with minibatch of size one day

Figure 2: Change in allocation over time. OpenAQ data.

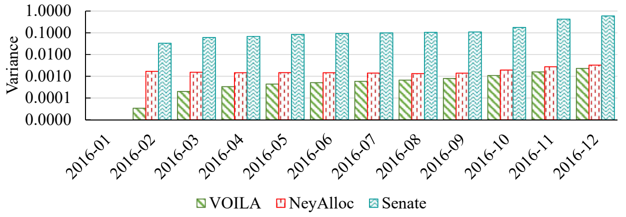


Figure 3: Variance of VOILA compared to NeyAlloc and Senate. Sample size: 1M records, OpenAQ data.

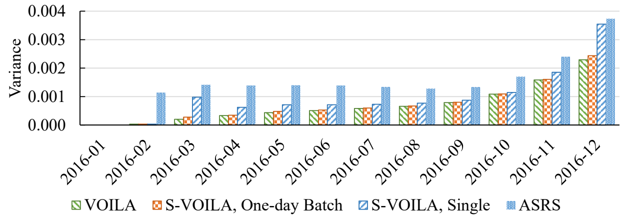


Figure 4: Variance of S-VOILA compared with ASRS and offline VOILA. Sample size 1M records, OpenAQ data.

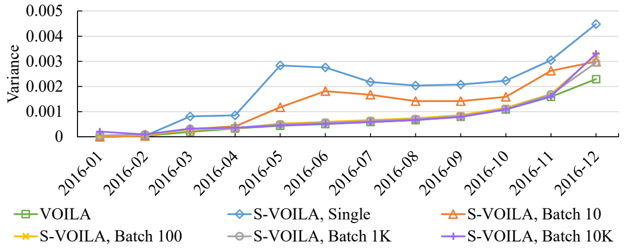


Figure 5: Impact of Minibatch Size on Variance, OpenAQ.

performance of S-VOILA improves. In November and December, new strata appear again, and the relative performance is again affected.

Among offline algorithms, we observe from Figure 3 that Senate performs poorly, since it blindly allocates equal space to all strata. NeyAlloc results in a variance that is larger than VOILA, by a factor of 1.4x to 50x. While NeyAlloc is known to be variance-optimal under the assumption of having all strata being abundant, these results show that it is far from variance-optimal for bounded strata.

Impact of Sample Size: To observe the sensitivity to the sample size, we conducted an experiment where the sample size is varied from 5000 to 1 million. We fixed the minibatch size to 100 thousand records. As expected, in both S-VOILA and VOILA, with single element and minibatch processing, the variance decreases when the sample size increases. The general trend was that the variance decreased by approximately a factor of 10 when the sample size increased by a factor of 10. We omit detailed results due to space constraints.

Impact of Minibatch Size: We further conducted an experiment where the minibatch size is chosen from $\{1, 10, 10^2, 10^3, 10^4\}$. The results are shown in Figure 5. A minibatch size of 10 elements yields significantly better results than single element S-VOILA. A minibatch size of 100 or greater makes the variance of S-VOILA nearly equal to the optimal variance.

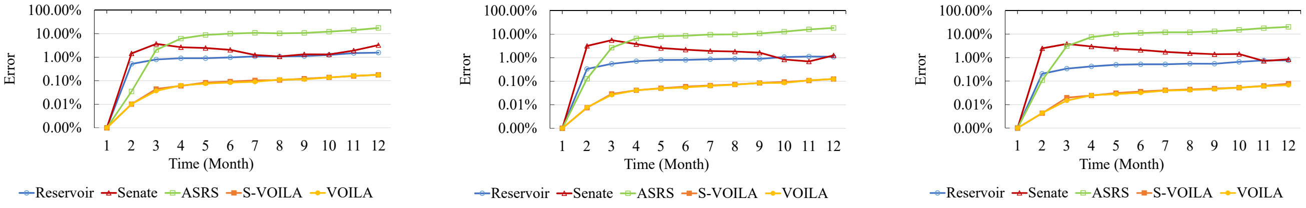
7.5 Query Performance, Infinite Window

We now evaluate the quality of these samples indirectly, through their use in approximate query processing. Samples constructed using S-VOILA and VOILA are used to approximately answer a variety of queries on the data so far. For evaluating the approximation error, we also implement an exact (but expensive) method for query processing Exact that stores all records in a MySQL database. Identical queries are made at the same time points in the stream to the different streaming and offline samplers, as well as to the exact query processor.

A range of queries are used. Each query *selects a subset of data through a selection predicate supplied at query time, and applies an aggregate.* This shows the flexibility of the sample, since it does not have any a priori knowledge of the selection predicate. We have chosen predicates with selectivity equal to one of 0.25, 0.50, and 1.00. We consider four aggregation functions: SUM, the sum of elements; SSQ, the sum of squares of elements; AVG, the mean of elements; and STD, the standard deviation. Each data point is the mean of five repetitions of the experiment with the same configuration. Each query was executed over all received data after one month of data arrived, up to entire year of 2016 in the OpenAQ dataset with thirty-one million records.

Figures 6 and 7 show the relative errors of different aggregations as the size of streaming data increases, while the sample size is held fixed. Both figures show that S-VOILA outperforms other streaming samplers across queries with different aggregation and selectivity. This result shows that S-VOILA maintains a better quality of stratified sample to answer an aggregation over a subset of data accurately. In addition, S-VOILA performs very closely to its offline version, VOILA, which samples from the entire received data. We note that when ASRS evicts elements from per-stratum samples, there may not always be new elements to take their place, hence it often does not use its full quota of allocated memory.

Alternate Methods of Stratification. We also experimented with the OpenAQ data set stratified in a different manner, using the city where the observation was made. Sample results are shown in Figure 8. We still see that S-VOILA outperforms

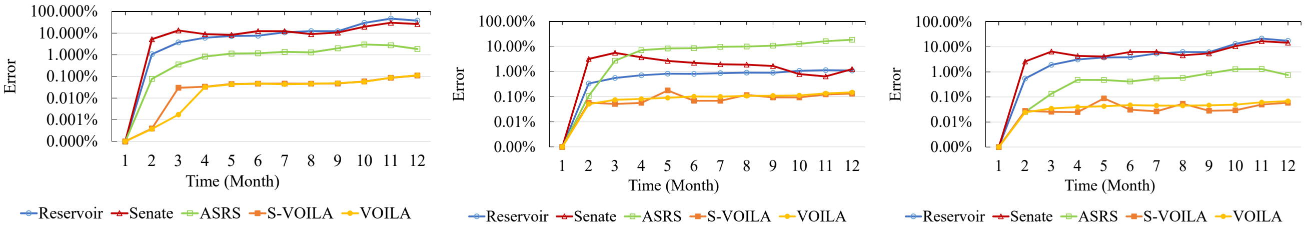


(a) Selectivity=0.25

(b) Selectivity=0.50

(c) Selectivity=1.00

Figure 6: Streaming samplers. SUM with different selectivities, sample size = 1 million. OpenAQ data.



(a) SSQ

(b) AVG

(c) STD

Figure 7: Streaming samplers. SSQ, AVG, and STD with selectivity 0.50, sample size = 1 million. OpenAQ data.

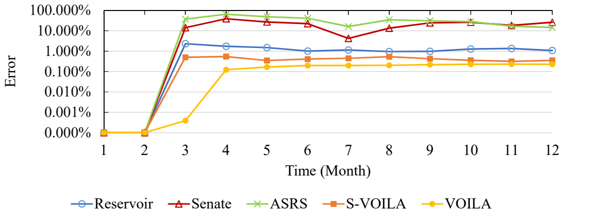


Figure 8: Streaming samplers, data stratified by the city (SUM with selectivity 0.5)

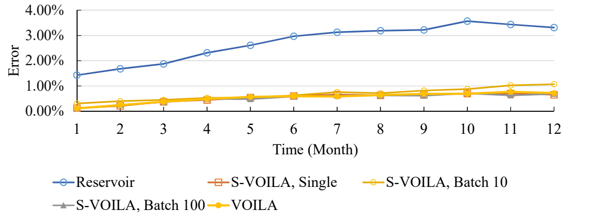


Figure 9: Streaming samplers, impact of minibatch size, sample size = 100,000. (SUM with selectivity 0.5)

Reservoir, Senate, and ASRS. This supports our observation that the sample maintained by S-VOILA is of a higher quality than other streaming samplers, no matter how data is stratified.

Impact of Sample Size. We also explored different sample sizes varied from 500, 000 to 1 million. All methods benefit from increased sample size and the relative performance between different methods remains the same across different sizes.

Impact of Minibatch Size. Figure 9 shows the impact of the minibatch size on the accuracy of streaming samplers for the SUM query with selectivity 0.5. The sample size is set to one hundred thousand for each sampler. S-VOILA with different minibatch sizes has an error less that 1%, often much smaller, while Reservoir has an error that is often 3% or larger. In addition, we observe that S-VOILA with different minibatch sizes is very close to VOILA.

7.6 Sliding Window Streaming

We experimented with streaming algorithms Reservoir and S-VOILA with a sliding window of size $W = 10^6$. The version of Reservoir that was used here maintains a uniform sample over the window by sampling each record with the same selection probability of $\frac{M}{W}$, so it may be more accurately termed "Bernoulli sampling". S-VOILA uses stratified sampling with single element processing, as described in Section 5. As the window slides, we periodically ask for sum of the *value* attribute in the current window. We report the error by compare the estimates from samples with the ground-truth answer. Figure 10 shows the average errors of 5 runs. With a 10% sample rate, as expected, S-VOILA provide an answer with less than 1% error, while Reservoir has an error of about 2-3%.

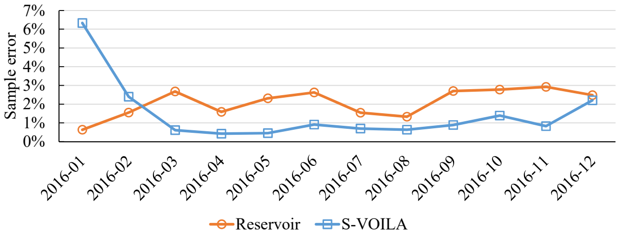
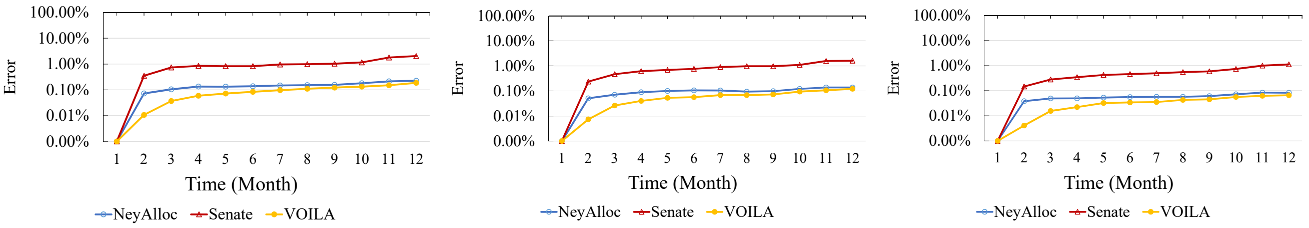


Figure 10: Sample error of window sum query, streaming data with sliding window of 10^6 and sample size of 10^5 records, OpenAQ data.

7.7 Offline Sampling

We also compared VOILA with other offline samplers for the SUM query with different selectivities. Figure 11 shows that VOILA always has better performance than Senate and NeyAlloc. Our experiments with other aggregations also showed similar results.



(a) Selectivity=0.25

(b) Selectivity=0.50

(c) Selectivity=1.00

Figure 11: Offline samplers. SUM with different selectivities, sample size = 1 million. OpenAQ data.

8 CONCLUSIONS

We presented S-VOILA, an algorithm for streaming SRS with minibatch processing, which interleaves a continuous, locally variance-optimal re-allocation of sample sizes with streaming sampling. Our experiments show that S-VOILA results in variance that is typically close to VOILA, which was given the entire input beforehand, and which is much smaller than that of algorithms due to prior work. We also show an inherent lower bound on the worst-case variance of any streaming algorithm for SRS – this limitation is not due to the inability to compute the optimal sample allocation in a streaming manner, but is instead due to the inability to increase sample sizes in a streaming manner, while maintaining uniformly weighted sampling within a stratum. Our work also led to a variance-optimal method VOILA for offline SRS from data that may have bounded strata. Our experiments show that on real and synthetic data, an SRS obtained using VOILA can have a significantly smaller variance than one obtained by Neyman allocation.

There are several directions for future research, including (1) restratification in a streaming manner, (2) handling group-by queries and join queries, (3) incorporating general versions of time-decay, and (4) SRS on distributed data.

Acknowlegdment: Nguyen and Tirthapura were supported in part by NSF grants 1527541 and 1725702.

REFERENCES

[1] S. Acharya, P. Gibbons, and V. Poosala. 2000. Congressional Samples for Approximate Answering of Group-by Queries. In *Proc. SIGMOD*. 487–498.

[2] S. Acharya, P. B. Gibbons, V. Poosala, and S. Ramaswamy. 1999. The Aqua Approximate Query Answering System. In *Proc. SIGMOD*. 574–576.

[3] S. Agarwal, B. Mozafari, A. Panda, H. Milner, S. Madden, and I. Stoica. 2013. BlinkDB: Queries with Bounded Errors and Bounded Response Times on Very Large Data. In *Proc. EuroSys*. 29–42.

[4] M. Al-Kateb and B. S. Lee. 2010. Stratified Reservoir Sampling over Heterogeneous Data Streams. In *Proc. SSDBM*. 621–639.

[5] M. Al-Kateb and B. S. Lee. 2014. Adaptive stratified reservoir sampling over heterogeneous data streams. *Information Systems* 39 (2014), 199–216.

[6] M. Al-Kateb, B. S. Lee, and X. S. Wang. 2007. Adaptive-Size Reservoir Sampling over Data Streams. In *Proc. SSDBM*. 22.

[7] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. 2002. Models and Issues in Data Stream Systems. In *Proc. PODS*. 1–16.

[8] B. Babcock, S. Chaudhuri, and G. Das. 2003. Dynamic Sample Selection for Approximate Query Processing. In *Proc. SIGMOD*. 539–550.

[9] B. Babcock, M. Datar, and R. Motwani. 2002. Sampling from a Moving Window over Streaming Data. In *SODA*.

[10] M. D Bankier. 1988. Power allocations: determining sample sizes for subnational areas. *The American Statistician* 42, 3 (1988), 174–177.

[11] V. Braverman, R. Ostrovsky, and G. Vorsanger. 2015. Weighted Sampling Without Replacement from Data Streams. *Inf. Process. Lett.* 115, 12 (2015), 923–926.

[12] V. Braverman, R. Ostrovsky, and C. Zaniolo. 2009. Optimal Sampling from Sliding Windows. In *Proc. PODS*. 147–156.

[13] V. Braverman, R. Ostrovsky, and C. Zaniolo. 2009. Optimal Sampling from Sliding Windows. In *PODS*.

[14] S. Chaudhuri, G. Das, and V. Narasayya. 2007. Optimized Stratified Sampling for Approximate Query Processing. *ACM TODS* 32, 2 (2007).

[15] Y. Chung and S. Tirthapura. 2015. Distinct Random Sampling from a Distributed Stream. In *IPDPS*. 532–541.

[16] Y. Chung, S. Tirthapura, and D. Woodruff. 2016. A Simple Message-Optimal Algorithm for Random Sampling from a Distributed Stream. *IEEE TKDE* 28, 6 (2016), 1356–1368.

[17] W. G. Cochran. 1977. *Sampling Techniques* (third ed.). John Wiley & Sons, New York.

[18] G. Cormode, S. Muthukrishnan, K. Yi, and Q. Zhang. 2012. Continuous Sampling from Distributed Streams. *JACM* 59, 2 (2012).

[19] G. Cormode, V. Shkapenyuk, D. Srivastava, and B. Xu. 2009. Forward Decay: A Practical Time Decay Model for Streaming Systems. In *Proc. ICDE*. 138–149.

[20] G. Cormode, S. Tirthapura, and B. Xu. 2009. Time-decaying Sketches for Robust Aggregation of Sensor Data. *SIAM J. Comput.* 39, 4 (2009), 1309–1339.

[21] M. Datar, A. Gionis, P. Indyk, and R. Motwani. 2002. Maintaining stream statistics over sliding windows. *SIAM J. Comput.* 31, 6 (2002), 1794–1813.

[22] P. S. Efraimidis and P. G. Spirakis. 2006. Weighted Random Sampling with a Reservoir. *Inf. Process. Lett.* 97, 5 (2006), 181–185.

[23] R. Gemulla and W. Lehner. 2008. Sampling Time-based Sliding Windows in Bounded Space. In *Proc. SIGMOD*. 379–392.

[24] R. Gemulla and W. Lehner. 2008. Sampling Time-based Sliding Windows in Bounded Space. In *SIGMOD*.

[25] R. Gemulla, W. Lehner, and P. J. Haas. 2008. Maintaining Bounded-size Sample Synopses of Evolving Datasets. *The VLDB Journal* 17, 2 (2008), 173–201.

[26] P. B. Gibbons and S. Tirthapura. 2001. Estimating Simple Functions on the Union of Data Streams. In *Proc. SPAA*. 281–291.

[27] P. B. Gibbons and S. Tirthapura. 2002. Distributed streams algorithms for sliding windows. In *SPAA*. 63–72.

[28] P. J. Haas. 2016. Data-Stream Sampling: Basic Techniques and Results. In *Data Stream Management*. Springer, 13–44.

[29] T. Johnson and V. Shkapenyuk. 2015. Data Stream Warehousing In Tidalrace. In *Proc. CIDR*.

[30] S. Joshi and C. Jermaine. 2008. Robust Stratified Sampling Plans for Low Selectivity Queries. In *Proc. ICDE*. 199–208.

[31] S. Kandula, A. Shanbhag, A. Vitorovic, M. Olma, R. Grandl, S. Chaudhuri, and B. Ding. 2016. Quickr: Lazily Approximating Complex AdHoc Queries in BigData Clusters. In *SIGMOD*. 631–646.

[32] K. Lang, E. Liberty, and K. Shmakov. 2016. Stratified Sampling Meets Machine Learning. In *Proc. ICML*. 2320–2329.

[33] S. L. Lohr. 2009. *Sampling: Design and Analysis* (2nd ed.). Duxbury Press.

[34] I. Mcleod and D. Bellhouse. 1983. A Convenient Algorithm for Drawing a Simple Random Sample. *Journal of the Royal Statistical Society. Series C. Applied Statistics* 32 (1983), 182–184.

[35] X. Meng. 2013. Scalable Simple Random Sampling and Stratified Sampling. In *Proc. ICML*. 531–539.

[36] J. Neyman. 1934. On the Two Different Aspects of the Representative Method: The Method of Stratified Sampling and the Method of Purposive Selection. *Journal of the Royal Statistical Society* 97, 4 (1934), 558–625.

[37] OpenAQ [n. d.]. http://openaq.org. ([n. d.]).

[38] S. K. Thompson. 2012. *Sampling* (3rd ed.). Wiley.

[39] Y. Tillé. 2006. *Sampling Algorithms* (1st ed.). Springer-Verlag.

[40] S. Tirthapura and D. P Woodruff. 2011. Optimal random sampling from distributed streams revisited. In *DISC*. 283–297.

[41] J. S. Vitter. 1983. Optimum Algorithms for Two Random Sampling Problems. In *Proc. FOCS*. 65–75.

[42] M. Zaharia, T. Das, H. Li, T. Hunter, S. Shenker, and I. Stoica. 2013. Discretized streams: fault-tolerant streaming computation at scale. In *SOSP*. 423–438.

[43] L. Zhang and Y. Guan. 2007. Variance estimation over sliding windows. In *PODS*. 225–232.