# Approximate Counts and Quantiles over Sliding Windows

**Arvind Arasu**
Stanford University
arvinda@cs.stanford.edu

**Gurmeet Singh Manku**
Stanford University
manku@cs.stanford.edu

**Abstract**

We consider the problem of maintaining approximate counts and quantiles over fixed- and variable-size sliding windows in limited space. For quantiles, we present deterministic algorithms whose space requirements are $O(\frac{1}{\epsilon}\log\frac{1}{\epsilon}\log N)$ and $O(\frac{1}{\epsilon}\log\frac{1}{\epsilon}\log\epsilon N\log N)$ in the worst-case for fixed- and variable-size windows, respectively, where $N$ denotes the current number of elements in the window. Our space bounds improve upon the previous best bounds of $O(\frac{1}{\epsilon^2}\ \text{polylog}(\frac{1}{\epsilon}, N))$.

For counts, we present both deterministic and randomized algorithms. The deterministic algorithms require $O(\frac{1}{\epsilon}\log^2\frac{1}{\epsilon})$ and $O(\frac{1}{\epsilon}\log^2\frac{1}{\epsilon}\log\epsilon N)$ worst-case space for fixed- and variable-size windows, respectively, while the randomized ones require $O(\frac{1}{\epsilon}\log\frac{1}{\epsilon\delta})$ and $O(\frac{1}{\epsilon}\log\frac{1}{\epsilon\delta}\log\epsilon N)$ worst-case space. We believe no previous work on space-efficient approximate counts for sliding windows exists.

## 1 Introduction

Data streams arise in several application domains like high-speed networking, finance, and transaction logs. For many applications, the recent stream elements are more important than those that arrived a long time ago. This preference for recent elements is commonly expressed using a *sliding window* [BBD+02], which is a portion of the stream between "now" and some time in the past. This paper considers the problem of *approximate*, *space-efficient* computation of two important statistics, counts and quantiles, over sliding windows. An exact answer for either statistic requires us to store the entire current-window, which is infeasible for many practical stream rates and window sizes.

## 2 Definitions

**Approximate Counts:** An $\epsilon$-*approximate count* for a bag[1] of $N$ elements is a set of $\langle$element, count$\rangle$ pairs such that (a) the count associated with an element is smaller than the *frequency* of the element (number of occurrences in the bag), but by at most $\epsilon N$, and (b) any element whose frequency exceeds $\epsilon N$ belongs to the set; other elements may or may not.

Approximate counts are important because they can be used to solve an approximate version of frequent elements problem [MM02], which has applications in domains like data mining and network monitoring (see Section 3). The frequent elements problem seeks the set of elements whose frequency exceeds $sN$ for a given threshold parameter $s$. The $\epsilon$-approximate version allows some false positives—elements whose frequency is greater than $(s-\epsilon)$ can appear in the answer—but no false negatives, i.e., all elements whose frequency is greater than $sN$ must appear in the answer. It is easily verified that an $\epsilon$-approximate count can be used to find $\epsilon$-approximate frequent elements if $s \geq \epsilon$.

**Approximate Quantiles:** Consider a bag of $N$ elements. The *rank* of an element is its position in a sorted arrangement of elements of the bag. The rank of the minimum element is 1, while that of the maximum element is $N$. The $\phi$-*quantile* of the bag is the element with rank $\lceil \phi N \rceil$. The 0.5-quantile is called the *median*. An element is said to be an $\epsilon$-*approximate* quantile if its rank is between $\lceil (\phi-\epsilon)N \rceil$ and $\lceil (\phi+\epsilon)N \rceil$; clearly there could be many elements that qualify.

[1] A bag is simply a multi-set, allowing multiple occurrences of an element.

**Sliding Windows**: A sliding window over a stream is a bag of last $N$ elements of the stream for some nonnegative integer $N$. We distinguish two variants of sliding windows based on whether $N$ is fixed (*fixed-size* sliding windows) or variable (*variable-size* sliding windows). Both variants can be modeled using an adversary who has the ability to insert a new element to the window or delete the oldest element from the window, at each step. For fixed-size windows, the adversary is constrained to perform the insertions and deletions in pairs, except in the beginning when (s)he has to insert exactly $N$ elements without a deletion. For variable-size windows the adversary has no constraints.

Fixed-size and variable-size windows model several variants of sliding windows considered in previous work. Tuple-based windows exactly correspond to fixed-size windows, while time-based windows (window of elements with last $T$ timestamps) can be modeled in a straightforward manner using variable-size windows. For approximate statistics with space as the primary consideration, variable-windows also model the $n$-of-$N$ computations of [LLXY04]. In the $n$-of-$N$ model, an algorithm should be able to compute approximate statistics for all windows of size $n$, smaller than a pre-specified constant $N$. An algorithm for variable-size windows necessarily has this ability, since an adversary can shrink a window by an arbitrary amount using a sequence of deletions.

## 3 Previous Work

### Frequency Counting Algorithms

For data streams, the earliest deterministic algorithm for $\epsilon$-approximate frequency counts is by Misra and Gries [MG82]. Their algorithm requires $\frac{1}{\epsilon}$ space and $O(1)$ amortized processing time per element. The same algorithm has been re-discovered recently by Demaine et al [DLOM02] and Karp et al [KSP03], who reduced the processing time to $O(1)$ in the worst case. Manku and Motwani [MM02] presented LOSSY COUNTING, a new deterministic algorithm that requires $O(\frac{1}{\epsilon}\log \epsilon N)$ space. Though the space requirements are worse than the Misra-Gries algorithm, LOSSY COUNTING is superior when the input is skewed. Further, the algorithm can be adapted to compute association rules over data streams.

In a random sample of size $O(\frac{1}{\epsilon^2}\log \delta^{-1})$, the relative frequency of any element in the sample differs from its true frequency in the base dataset by at most $\epsilon$. This observation can be exploited to obtain approximate frequency counts in a single pass such that the space requirements are independent of $N$. For example, Toivonen [Toi96] identifies a candidate set of frequent itemsets in the context of association rule mining [AIS93]. For data streams, Manku and Motwani [MM02] presented STICKY SAMPLING, a randomized algorithm that requires only $O(\frac{1}{\epsilon}\log \delta^{-1})$ space, beating the sampling bound. Cormode and Muthukrishnan [CM03] recently presented randomized algorithms for identifying frequency counts in the presence of both additions and deletions. Their algorithm is not directly applicable to sliding windows where the oldest element is *implicitly* deleted.

In this paper, we propose deterministic and randomized algorithms for approximate frequency counts over sliding windows. Our randomized algorithms are adaptations of STICKY SAMPLING. For fixed-size and variable-size windows, we require $O(\frac{1}{\epsilon}\log \delta^{-1})$ and $O(\frac{1}{\epsilon}\log \delta^{-1}\log \epsilon N)$ space respectively. Our deterministic algorithms use the Misra-Gries algorithm as a black-box. For fixed-size and variable-size windows, we require $O(\frac{1}{\epsilon}\log^2 \frac{1}{\epsilon})$ and $O(\frac{1}{\epsilon}\log^2 \frac{1}{\epsilon}\log \epsilon N)$ space respectively.

### Deterministic Quantile-Finding Algorithms

The history of quantile-finding algorithms dates back to the early days of computer science. Early work focused on main memory datasets. The celebrated paper of Blum, Floyd, Pratt, Rivest and Tarjan[BFP$^+$73], shows that selecting the $k$th largest element among $N$ elements requires at least $1.5N$ and at most $5.43N$ comparisons. For an account of progress since then, see the survey by Paterson [Pat96].

For large datasets in external memory, Pohl [Poh69] established that any deterministic algorithm that computes the exact median in one pass needs to store at least $N/2$ data elements. Munro and Paterson [MP80] generalized the idea and showed that for $p \geq 2$, memory to store $\Theta(N^{1/p})$ elements is required

for finding the exact median (or any $\phi$-quantile) in $p$ passes.

For data streams, where only one pass is allowed, the lower bound of $N/2$ for the exact median [Poh69] motivated the definition of *approximate* quantiles in the hope of reducing space to $o(N)$. Manku et al [MRL98] defined the notion of $\epsilon$-approximate $\phi$-quantiles. Building upon the Munro-Paterson paper, they devised a deterministic one pass algorithm that requires only $O(\frac{1}{\epsilon}\log^2 \epsilon N)$ space. However, the algorithm requires advance knowledge of an upper bound for $N$ and therefore does not work for infinitely long streams. Recently, Greenwald and Khanna [GK01] improved the space requirements to $O(\frac{1}{\epsilon}\log \epsilon N)$, without requiring advance knowledge of $N$. Empirically, their algorithm has been observed to require only $O(\frac{1}{\epsilon})$ space if the input is a random permutation of elements.

For sliding windows, Lin et al [LLXY04] recently devised the first algorithms for $\epsilon$-approximate quantiles. Their space bounds are $O(\frac{1}{\epsilon^2}+\frac{1}{\epsilon}\log(\epsilon^2 N))$ for fixed-size windows and $O(\frac{1}{\epsilon^2}\log^2(\epsilon N))$ for variable-size windows. In this paper, we improve upon both bounds, our space requirements being $O(\frac{1}{\epsilon}\log\frac{1}{\epsilon}\log N)$ and $O(\frac{1}{\epsilon}\log\frac{1}{\epsilon}\log \epsilon N\log N)$ respectively.

**Randomized Quantile-Finding Algorithms**

For identifying exact quantiles in main memory, a simple linear time randomized algorithm was presented by Floyd and Rivest [FR75]. For approximate quantiles, randomization reduces the space requirements significantly. The key insight is the well-known fact that the $\phi$-quantile of a random sample of size $O(\frac{1}{\epsilon^2}\log\delta^{-1})$ is an $\epsilon$-approximate $\phi$-quantile of $N$ elements with probability at least $1-\delta$. This observation has been exploited by DeWitt *et al* [DNS91] to identify splitters of large datasets in the context of distributed sorting. For data streams, a randomized quantile-finding algorithm was proposed by Manku et al [MRL99] that requires only $O(\frac{1}{\epsilon}\log^2(\frac{1}{\epsilon}\log(\epsilon\delta)^{-1}))$ space, beating the sampling bound. Further improvement in space is possible, as described in Section 5.

**Related Problems**

A problem related to approximate counting is the top-$k$ problem, also known as the *Hot Item* problem, where the goal is to identify $k$ items which are most frequent. Algorithms for the problem over data streams have been developed by Charikar et al [CCFC02], Cormode and Muthukrishnan [CM03] and Gibbons and Matias [GM98].

Sliding window algorithms have been developed for a variety of problems: bit-counting (Datar et al [DGIM02]), sampling (Babcock et al [BDM02]), variance and $k$-medians (Datar et al [BDMO03], distinct values and bit-counts (Gibbons and Tirthapura [GT02]) and quantiles (Lin et al [LLXY04]).

## 4 Deterministic Algorithms for Fixed-size Windows

We now present deterministic algorithms for $\epsilon$-approximate counts and quantiles for a fixed-size sliding window containing $N$ elements. We assume both $1/\epsilon$ and $N$ are powers of 2 to avoid floors and ceilings in expressions; generalization to arbitrary $\epsilon$ and $N$ is straightforward. Our presentation focuses on the state maintained by the algorithms, which we call a *sketch*. In particular, we describe how a sketch is maintained and how an approximate statistic (count/quantile) is computed using it. For both counts and quantiles, we denote a (deterministic) sketch for a window of size $N$ and error $\epsilon$ as $\mathcal{F}(N,\epsilon)$.

We begin by defining *blocks* and *levels* (see Figure 1). Levels are numbered sequentially $0,1,2,\ldots$. For each level, we associate a partitioning of the stream into non-overlapping equi-sized blocks. The size of blocks in level-0 is $\frac{\epsilon N}{4}$, and the size doubles from one level to the next higher one. So, the size of blocks in level-$\ell$ is $\frac{\epsilon N}{4}2^\ell$. Within a level, blocks are numbered $0,1,2,\ldots$; smaller numbered blocks contain older elements. For example, in level-0, block 0 contains the first $\frac{\epsilon N}{4}$ elements, block 1 the next $\frac{\epsilon N}{4}$, and so on. In general, block $b$ in level-$\ell$ contains elements with positions in the range $[b2^\ell\frac{\epsilon N}{4}, (b+1)2^\ell\frac{\epsilon N}{4}-1]$.

At any point of time, we assign one of 4 states to a block: the state is *active* if all its elements belong to the current window, *expired* if atleast one of its elements is older than the last $N$ elements, *under*

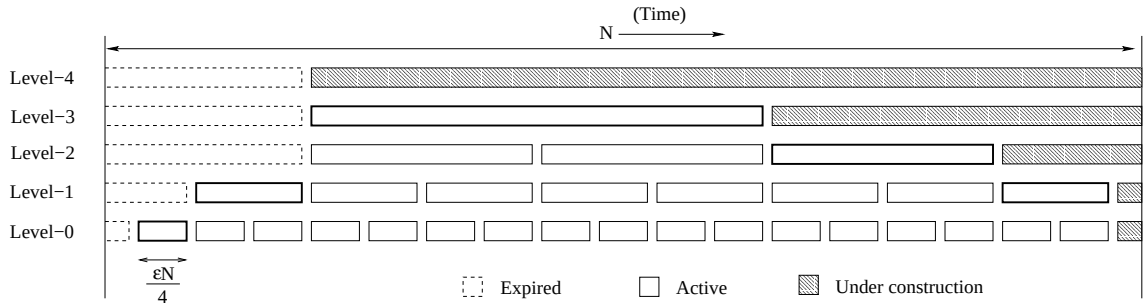


*Figure 1:* Levels and Blocks used in $\mathcal{F}(N, \epsilon)$

*construction* if some of its elements belong to the current window, and all the remaining, yet to arrive, and *inactive* if none of its elements has arrived. All blocks are initially in inactive state and eventually reach expired state. At any given point of time, there are about $\frac{4}{\epsilon}$ active level-0 blocks, $\frac{2}{\epsilon}$ active level-1 blocks, and so on. The highest level at which a block is active or under construction is $\log_2 \frac{4}{\epsilon}$; blocks of higher levels pass directly from inactive state to expired state.

$\mathcal{F}(N, \epsilon)$ is a collection of sketches, one per active block. A sketch for a block is a synopsis that allows retrieval of approximate statistics (count/quantile) of elements belonging to the block. A sketch for counts is different from a sketch for quantiles; we present details in Sections 4.1 and 4.2. When an approximate statistic is required over the current window, the sketches of active blocks are merged to determine the answer. Again, the merge operation for count sketches and quantile sketches are different, and we present details later. In order to construct the sketch for a block, when the block is under construction, we run a traditional one-pass algorithm (Misra-Gries algorithm for counts and Greenwald-Khanna algorithm for quantiles) with an appropriate error parameter, over elements of the block. When the block eventually becomes active, the data structure maintained by the one-pass algorithm is probed to construct a sketch for the block. In the definition of $\mathcal{F}(N, \epsilon)$ below, let $L = \log_2(\frac{4}{\epsilon})$ denote the highest level at which an active block exists.

> $\mathcal{F}(N, \epsilon)$: For each level-$\ell$ block currently in active state, store an $\epsilon_\ell$-sketch (count/quantile) of the elements of the block, where $\epsilon_\ell = \frac{\epsilon}{2(2L+2)} 2^{(L-\ell)}$. For the level-$\ell$ block currently under construction, store the state required by the one-pass algorithm being executed (Misra-Gries / Greenwald-Khanna) to compute an $\epsilon_\ell$-sketch for the block.

### 4.1 Counts

For counts, an $\epsilon_\ell$-sketch for a block is just an $\epsilon_\ell$-approximate count for the block, i.e., the sketch is a set of ⟨element, count⟩ pairs such that ($a$) the count associated with an element is smaller than its frequency in the block, but by atmost $\epsilon_\ell M$ (where $M$ is the size of the block), and ($b$) any element whose frequency in the block is greater than $\epsilon_\ell M$ appears in the set. For a level-$\ell$ block, we obtain an $\epsilon_\ell$-sketch of the block that uses $O(\frac{1}{\epsilon_\ell})$ space by running Misra-Gries algorithm [MG82] over the block when it is under construction, and storing the output of the algorithm when the block becomes active. The construction algorithm itself requires $O(\frac{1}{\epsilon_\ell})$ space. The next Lemma shows how sketches over disjoint bags can be merged to produce an approximate count for the union of the bags.

**Lemma 4.1** *(Approximate Counts) Using a collection of $s$ sketches over disjoint bags of $N_1, N_2, \ldots N_s$ elements each, with error parameters $\epsilon_1, \epsilon_2, \ldots \epsilon_s$, respectively, we can compute an approximate count with error parameter $\frac{\epsilon_1 N_1 + \epsilon_2 N_2 + \cdots + \epsilon_s N_s}{N_1 + N_2 + \cdots + N_s}$ for the union of the bags.*

**Proof:** The approximate count is obtained by summing all the counts associated with an element that occurs in one or more of the $s$ sketches. (Recall that each sketch contains a set of ⟨element, count⟩ pairs.)

Clearly, the approximate count enjoys the property that the sum of counts of an element is below the frequency of the element in the bag-union by at most $\epsilon_1 N_1 + \epsilon_2 N_2 + \cdots + \epsilon_s N_s$. Moreover, any element whose cumulative frequency exceeds $\epsilon_1 N_1 + \epsilon_2 N_2 + \cdots + \epsilon_s N_s$ belongs to at least one of the $s$ sketches, and therefore appears in the approximate count. □

**Theorem 4.1** *$\mathcal{F}(N, \epsilon)$ allows $\epsilon$-approximate counts to be computed over the last $N$ elements, and uses $O(\frac{1}{\epsilon}\log^2\frac{1}{\epsilon})$ space.*

**Proof:** Let $L = \log_2(\frac{4}{\epsilon})$ denote the highest level at which an active block exists. For level-$\ell$ blocks, $\mathcal{F}(N, \epsilon)$ stores $\epsilon_\ell$-sketches with $\epsilon_\ell = \frac{\epsilon}{2(2L+2)}2^{(L-\ell)}$. Let $N_\ell = 2^\ell(\epsilon N/4)$ denote the size of a level-$\ell$ block. Then $\epsilon_\ell N_\ell = \frac{\epsilon}{2(2+2L)}2^L(\epsilon N/4) = \frac{\epsilon N}{2(2+2L)}$, which is independent of $\ell$.

For computing approximate counts, we only consider elements that belong to some active block, and ignore the rest. The elements in the current window that we ignore belong to either the level-0 block that expired most recently or the level-0 block that is under construction; there are at most $(\frac{\epsilon N}{4} - 1)$ elements in each, and so we ignore at most $2(\frac{\epsilon N}{4} - 1)$ elements. We claim that the remaining $\tilde{N} \geq N - 2(\frac{\epsilon N}{4} - 1)$ elements of the window can be expressed as a union of at most $2L + 2$ nonoverlapping blocks, such that at most two blocks belong to any level (see Figure 1). Using Lemma 4.1, sketches for these blocks can be merged to produce approximate counts with error parameter $\epsilon$ such that $\epsilon\tilde{N} \leq \sum_{\ell=0}^{L} 2\epsilon_\ell N_\ell = \frac{2(1+L)\epsilon N}{2(2L+2)} = \epsilon N/2$. The total absolute error in counts is bounded by $\epsilon\tilde{N} + 2(\epsilon N/4) - 2$, which is no more than $\epsilon N$.

An $\epsilon_\ell$-sketch for a level-$\ell$ block uses $O(\frac{1}{\epsilon_\ell})$ space. There are at most $2^{L-\ell}$ active blocks at level-$\ell$. Thus the total space requirements for the sketches of active blocks is at most $\sum_{\ell=0}^{L} 2^{L-\ell}/\epsilon_\ell$. Substituting $\epsilon_\ell = \frac{\epsilon}{2(2L+2)}2^{(L-\ell)}$, we obtain $\sum_{\ell=0}^{L}\frac{1}{\epsilon}(4 + 4\log(4/\epsilon)) = O(\frac{1}{\epsilon}\log^2\frac{1}{\epsilon})$. The space requirements for running the Misra-Gries algorithm over blocks that are under construction is at most $\sum_{\ell=0}^{L} 1/\epsilon_\ell = O(\frac{1}{\epsilon}\log\frac{1}{\epsilon})$. The overall space is thus $O(\frac{1}{\epsilon}\log^2\frac{1}{\epsilon})$. □

### 4.2 Quantiles

For a level-$\ell$ block under construction, we run the Greenwald-Khanna algorithm [GK01] over the elements of the block (as they arrive) with error parameter $\frac{\epsilon_\ell}{2}$, where $\epsilon_\ell = \frac{\epsilon}{2(2L+2)}2^{(L-\ell)}$ is the desired error parameter for a sketch of a level-$\ell$ block. When the block become active, we retrieve $\phi$-quantiles for $\phi = \langle\epsilon_\ell, 2\epsilon_\ell, \ldots, 1\rangle$ and store the resulting sequence as the sketch. It is known [GK01] that this results in an $\epsilon_\ell$-sketch for the block, and clearly the sketch uses $O(\frac{1}{\epsilon_\ell})$ space. Further, the Greenwald-Khanna algorithm itself requires $O(\frac{1}{\epsilon_\ell}\log\epsilon_\ell N_\ell)$ space, where $N_\ell$ denotes the size of a level-$\ell$ block.

**Lemma 4.2** *(Approximate Quantiles) Using a collection of $s$ sketches over disjoint bags of $N_1, N_2, \ldots N_s$ elements each, with error parameters $\epsilon_1, \epsilon_2, \ldots \epsilon_s$, respectively, we can compute $\epsilon$-approximate quantiles with error parameter $\epsilon = \frac{\epsilon_1 N_1 + \epsilon_2 N_2 + \cdots + \epsilon_s N_s}{N_1 + N_2 + \cdots + N_s}$ for the union of the bags.*

**Proof:** For simplicity, we assume that $1/\epsilon_i$ $(1 \leq i \leq s)$ is an integer. Also, assume that the sketches are constructed as described earlier, i.e., the $\epsilon_i$-sketch for the $i^{\text{th}}$ bag contains $\frac{\epsilon_i}{2}$-approximate quantiles for $\phi = \langle\epsilon_i, 2\epsilon_i, \ldots, 1\rangle$, though the lemma is valid for the more general class of sketches introduced in [GK01]. Associate a weight of $\epsilon_i N_i$ with each element of the $i$-th sketch. Sort the elements of all sketches and pick the element with the property that the sum of weights of all preceding elements is $< \lceil\phi(N_1 + \ldots + N_s)\rceil$, but the sum including the element's weight is $\geq \lceil\phi(N_1 + \ldots + N_s)\rceil$. We claim that this element is an $\epsilon$-approximate $\phi$-quantile for the union of the bags, where $\epsilon = \frac{\epsilon_1 N_1 + \epsilon_2 N_2 + \cdots + \epsilon_k N_k}{N_1 + N_2 + \cdots + N_k}$. □

**Theorem 4.2** *$\mathcal{F}(N, \epsilon)$ allows $\epsilon$-approximate quantiles to be computed over the last $N$ elements, and uses $O(\frac{1}{\epsilon}\log\frac{1}{\epsilon}\log N)$ space.*

**Proof:** The proof that $\mathcal{F}(N,\epsilon)$ allows $\epsilon$-approximate quantile computation is identical to the corresponding proof for counts, if we use Lemma 4.2 instead of Lemma 4.1. The space required for an $\epsilon_\ell$ quantile sketch is $O(\frac{1}{\epsilon_\ell})$, which is the same as that for an $\epsilon_\ell$ count sketch. So, it follows from the proof of Theorem 4.1 that the space required for the sketches of active blocks is $O(\frac{1}{\epsilon}\log^2\frac{1}{\epsilon})$. As we noted earlier, the space used by Greenwald-Khanna [GK01] algorithm to construct a sketch of level-$\ell$ block is $O(\frac{1}{\epsilon_\ell}\log\epsilon_\ell N_\ell)$, where $\epsilon_\ell = \frac{\epsilon}{2(2L+2)}2^{(L-\ell)}$ and $N_\ell = 2^\ell(\epsilon N/4)$ (size of the block). The total space over all levels is $\sum_{\ell=0}^{L} O(\frac{1}{\epsilon_\ell}\log\epsilon_\ell N_\ell)$, where $L = \log_2\frac{4}{\epsilon}$ is the maximum number of levels. We know that from proof of Theorem 4.1 that $\epsilon_\ell N_\ell = \frac{\epsilon N}{2(2L+2)}$ is independent of $\ell$. So, the above sum reduces to $(\log\frac{\epsilon N}{2(2L+2)})\sum_{\ell=0}^{L} O(\frac{1}{\epsilon_\ell})$. Since, $\epsilon_\ell \propto 2^{-\ell}$, $\sum_{\ell=0}^{L} O(\frac{1}{\epsilon_\ell})$ is geometric and is $O(\frac{1}{\epsilon_L}) = O(\frac{1}{\epsilon}\log\frac{1}{\epsilon})$. The total space used for constructing sketches is therefore $O(\frac{1}{\epsilon}\log\frac{1}{\epsilon}\log\frac{\epsilon N}{\log(1/\epsilon)})$. The overall space is $O(\frac{1}{\epsilon}\log^2\frac{1}{\epsilon} + \frac{1}{\epsilon}\log\frac{1}{\epsilon}\log\frac{\epsilon N}{\log(2/\epsilon)})$ which can be simplified to $O(\frac{1}{\epsilon}\log\frac{1}{\epsilon}\log N)$. □

## 5 Randomized Algorithms for Fixed-size Windows

Randomized algorithms for counts and quantiles can be developed based on the following facts: the $\phi$-quantile of a random sample of size $O(\frac{1}{\epsilon^2}\log\delta^{-1})$ is an $\epsilon$-approximate $\phi$-quantile of $N$ elements with probability at least $1-\delta$. Moreover, the relative frequency of any element within the sample differs from its true relative frequency by at most $\epsilon$, with probability at least $1-\delta$. However, the error in frequency counts is two-sided, unlike the one-sided error in our definition of $\epsilon$-approximate frequency count synopsis. We now show how the sampling bound can be beaten for both counts and quantiles.

### Counts

For maintaining approximate frequency counts over the last $N$ elements, we now present a variation of STICKY SAMPLING, an algorithm proposed earlier in [MM02] for data streams. Let $r$ denote an integer satisfying $1/2^r \leq (\frac{1}{\epsilon}\log(\epsilon\delta)^{-1})/N < 1/2^{r-1}$. We partition the stream into contiguous groups of $2^r$ elements each. We sample exactly 1 element per group, chosen uniformly at random. Corresponding to each sample, we maintain an ⟨element, count, position⟩ triple, where (i) count equals the number of occurrences of that element after the sample is created but before the same element is sampled again, and (ii) position denotes the position in the stream at which this sample was created. A triple is remembered as long as its position is within the last $N$ positions in the stream. An $\epsilon$-approximate frequency count sketch can be inferred from the set of triples in a natural way. Since the same element might be sampled multiple times, we simply sum the counts associated with all triples to which it belongs.

**Theorem 5.1** *$\epsilon$-approximate frequency counts can be maintained over the last $N$ elements in a data stream using $O(\frac{1}{\epsilon}\log(\epsilon\delta)^{-1})$ space, where $\delta$ denotes the probability of failure.*

**Proof:** The maximum number of elements sampled in a window of size $N$ is less than $N/2^{r-1} \leq \frac{2}{\epsilon}\log(\epsilon\delta)^{-1}$. This proves the space bound.

Consider an element whose frequency within the window exceeds $\epsilon N$. Let $k$ denote the number of groups in which its first $\epsilon N$ occurrences within the current window. The current window includes all these groups completely, with the possible exception of two groups: the oldest and the newest. Let $\epsilon N = c_1 + c_2 + \cdots + c_k$, where $c_1, c_2, \ldots, c_k$ denote the number of occurrences (that lie within the current window) in each of these $k$ groups. The probability that none of these occurrences is sampled equals $\Pi_{i=1}^{k}(1 - c_i/2^r)$. Note that partial overlap with the oldest and the newest group does not affect this expression. Since $1 - c_i/2^r < (1 - 1/2^r)^{c_i}$, the probability equals $\Pi_{i=1}^{k}(1 - 1/2^r)^{c_i} = (1 - 1/2^r)^{\epsilon N} < e^{-\epsilon N/2^r} < (\epsilon\delta)$. There are at most $1/\epsilon$ elements whose frequency count is more than $\epsilon N$ within the current window. Therefore, the probability that the count of any of them is below its true count by $\epsilon N$ or more, is at most $\delta$. □

Our version of STICKY SAMPLING is different from the one in [MM02] in two ways. First, we maintain approximate counts over sliding windows instead of streams. Second, our sampling scheme is different. In [MM02], each new element in the stream is sampled independently with probability $1/2^r$. Thus space requirements are $O(\frac{1}{\epsilon}\log(\epsilon\delta)^{-1})$ in expectation. With 1-in-$2^r$ sampling, as described above, the space requirements become worst-case.

**Quantiles**

For data streams, a sample of size $O(\frac{1}{\epsilon^2}\log\delta^{-1})$ can be maintained with Vitter's reservoir sampling [Vit85]. A faster algorithm is to select 1 out of $2^k$ successive elements with $k = \lfloor \log_2 N/(\frac{1}{\epsilon^2}\log\delta^{-1})\rfloor$, where $N$ denotes the current length of the stream. Thus $k$ grows logarithmically in proportion to the stream length. The error guarantees on the quantiles of the sample thus produced match the guarantees for samples produced by Vitter's scheme, as shown in [MRL99]. The $\frac{1}{\epsilon^2}$ term in the sample size can be problematic for tiny values of $\epsilon$. In practice, several samples are identical, allowing compression, as suggested by Gibbons and Matias [GM98]. If $N$ is known in advance, Manku et al [MRL98] show that further reduction in space is possible. The key idea is to feed the output of 1-out-of-$2^k$ sampling to a deterministic quantile-finding algorithm, distributing the error equally between the two steps. Using the algorithm by Greenwald and Khanna [GK01], we thus arrive at a randomized algorithm requiring only $O(\frac{1}{\epsilon}\log(\frac{1}{\epsilon}\log\delta^{-1}))$ space[2]. This yields the following theorem.

**Theorem 5.2** *$\epsilon$-approximate quantiles can be maintained over the last $N$ elements in a data stream using $O(\frac{1}{\epsilon}\log\frac{1}{\epsilon}\log(\frac{1}{\epsilon}\log(\epsilon\delta)^{-1}))$ space, where $\delta$ denotes the probability of failure.*

**Proof:** The overall algorithm is the same as the deterministic quantile-finding algorithm developed in Section 4, except that we change the algorithm used for blocks under construction. We replace the deterministic Greenwald-Khanna algorithm by the following randomized algorithm: We feed the output of 1-out-of-$2^k$ sampling described in [MRL99] to the Greenwald-Khanna algorithm, distributing the error equally between the two steps. Such a randomized algorithm requires only $O(\frac{1}{\epsilon}\log(\frac{1}{\epsilon}\log\frac{1}{\epsilon\delta}))$ space, independent of $N$ (see [MRL98] for a discussion of this idea). The sketches for active blocks require $O(\frac{1}{\epsilon}\log^2\frac{1}{\epsilon})$ space as before. However, blocks under construction now require only $O(\frac{1}{\epsilon}\log\frac{1}{\epsilon}\log(\frac{1}{\epsilon}\log(\epsilon\delta)^{-1}))$ space. The total space complexity is $O(\frac{1}{\epsilon}\log\frac{1}{\epsilon}\log(\frac{1}{\epsilon}\log(\epsilon\delta)^{-1}))$. □

## 6 Deterministic Algorithms for Variable-Size Windows

Variable-size windows were defined in Section 1. At every time-step, an adversary is allowed to either insert a new element into the window or delete the oldest element. The size of the window thus varies over time. We denote the current size of the window by $n$. Our goal is to maintain sketches for counts and quantiles over variable-size windows. For both the problems, we use $\mathcal{V}(n,\epsilon)$ to denote a (deterministic) sketch over the last $n$ elements in the stream, where $n$ varies over time. When a new element enters the window, the new sketch is $\mathcal{V}(n+1,\epsilon)$. When the oldest element leaves, the new sketch is $\mathcal{V}(n-1,\epsilon)$. For ease of exposition, we will assume that $1/\epsilon$ equals a power of two. If it is not, we can always scale $\epsilon$ down to $1/2^r$ where $1/2^r \le \epsilon < 1/2^{r-1}$, without affecting asymptotic space complexity.

For any $n$, $N$ and $\epsilon$, we define $\mathcal{F}_n(N,\epsilon)$ to be a *restriction* of the sketch $\mathcal{F}(N,\epsilon)$ to just the last $n$ elements. See Figure 2 for an illustration. The block sizes and error parameters associated with each level are the same as before. The only difference lies in the definition of active blocks. For $\mathcal{F}_n(N,\epsilon)$, only those blocks are said to be active all of whose elements belong to the $n$ most-recent elements (instead of the $N$ most-recent elements). For $n \ge N$, $\mathcal{F}_n(N,\epsilon)$ is identical to $\mathcal{F}(N,\epsilon)$.

[2]If $N$ is not known in advance, $k$ is not fixed but grows logarithmically with $N$. However, we claim without proof that it is possible to employ the adaptive sampling scheme in [MRL99] in conjunction with a modified Greenwald-Khanna algorithm to arrive at the same bound.

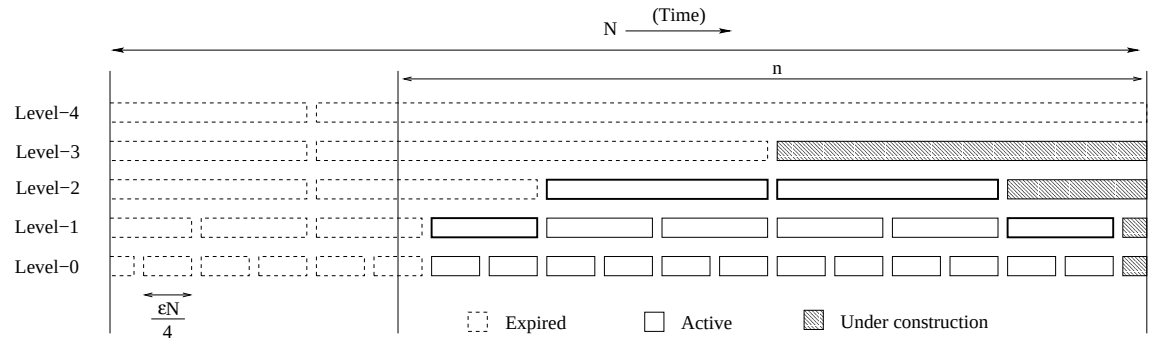


*Figure 2:* $\mathcal{F}_n(N,\epsilon)$, restriction of $\mathcal{F}(N,\epsilon)$ to last $n$ elements

We will soon describe a general technique for constructing $\mathcal{V}(n,\epsilon)$ from a collection of sketches $\mathcal{F}_n(N,\epsilon)$ treated as black-boxes, as long as $\mathcal{F}_n(N,\epsilon)$ satisfies the following two properties:

Property I: $\mathcal{F}_n(N,\epsilon)$ allows the computation of $\frac{\epsilon N}{n}$-approximate statistics over the last $n$ elements.

Property II: For any $k$ and $\epsilon$, $\mathcal{F}_{2^k}(2^{k+1},\epsilon)$ can be constructed from $\mathcal{F}_{2^k}(2^k,\epsilon)$.

$\mathcal{V}(n,\epsilon)$ is defined as follows: Let $k$ be an integer satisfying $2^{k-1} < n \leq 2^k$. Sketch $\mathcal{V}(n,\epsilon)$ is the collection of sketches $\{\mathcal{F}_n(2^k,\frac{\epsilon}{2}),\mathcal{F}_n(2^{k-1},\frac{\epsilon}{2}),\ldots,\mathcal{F}_n(\frac{2}{\epsilon},\frac{\epsilon}{2})\}$. We now show how $\mathcal{V}(n,\epsilon)$ is maintained in the face of additions and deletions of elements, and how $\epsilon$-approximate statistics can be retrieved for any $n' \leq n$.

**Deletion:** When the oldest element is deleted, we simply compute $\mathcal{F}_{n-1}(2^k,\frac{\epsilon}{2})$ from $\mathcal{F}_n(2^k,\frac{\epsilon}{2})$. If $n-1$ equals $2^{k-1}$, then sketch $\mathcal{F}_{2^{k-1}}(2^k,\frac{\epsilon}{2})$ is redundant since we already have $\mathcal{F}_{2^{k-1}}(2^{k-1},\frac{\epsilon}{2})$, which is sufficient; so sketch $\mathcal{F}_{2^{k-1}}(2^k,\frac{\epsilon}{2})$ is dropped.

**Addition:** Each of the sketches comprising $\mathcal{V}(n,\epsilon)$ is updated, resulting in the collection $\{\mathcal{F}_{n+1}(2^k,\frac{\epsilon}{2})$, $\mathcal{F}_{n+1}(2^{k-1},\frac{\epsilon}{2})$, …, $\mathcal{F}_{n+1}(\frac{2}{\epsilon},\frac{\epsilon}{2})\}$. If $n+1 = 2^k+1$, then we need to create the sketch $\mathcal{F}_{2^k+1}(2^{k+1},\frac{\epsilon}{2})$ and add it to the collection in order to create $\mathcal{V}(n+1,\epsilon)$. Property II ensures that $\mathcal{F}_{2^k}(2^{k+1},\frac{\epsilon}{2})$ can be constructed for counts and quantiles from $\mathcal{F}_{2^k}(2^k,\frac{\epsilon}{2})$. We then insert the newly arrived element into $\mathcal{F}_{2^k}(2^{k+1},\frac{\epsilon}{2})$ to create $\mathcal{F}_{2^k+1}(2^{k+1},\frac{\epsilon}{2})$.

**Retrieval:** To retrieve $\epsilon$-approximate statistics for any $n' \leq n$, we first identify integer $\ell$ satisfying $2^{\ell-1} < n' \leq 2^\ell$. Then we query sketch $\mathcal{F}_n(2^\ell,\frac{\epsilon}{2})$. Since $\frac{2^\ell(\epsilon/2)}{n'} < \epsilon$, Property I guarantees that $\epsilon$-approximate statistics over the last $n'$ elements can be retrieved from $\mathcal{F}_n(2^\ell,\frac{\epsilon}{2})$.

**Lemma 6.1** *Both properties are true for counts and quantiles sketches.*

**Proof:** (Property I) Using sketches in $\mathcal{F}(N,\epsilon)$ that correspond to active blocks restricted to the last $n$ elements, we can compute approximate counts and quantiles with an absolute error of $\epsilon N$. The relative error over $n$ elements is $\frac{\epsilon N}{n}$.

(Property II) The basic intuition is that $\mathcal{F}_{2^k}(2^k,\epsilon)$ is a "more accurate" sketch than $\mathcal{F}_{2^k}(2^{k+1},\epsilon)$: for blocks of a given size, $\mathcal{F}_{2^k}(2^k,\epsilon)$ maintains sketches with error parameter that is half as that of the sketches maintained by $\mathcal{F}_{2^k}(2^{k+1},\epsilon)$. Figure 3 illustrates the construction of $\mathcal{F}_{2^k}(2^{k+1},\epsilon)$ from $\mathcal{F}_{2^k}(2^k,\epsilon)$. For the formal proof, for simplicity, we assume that no block of $\mathcal{F}_{2^k}(2^k,\epsilon)$ is under construction (as indicated in Figure 3). This is equivalent to assuming that the current length of the stream is an exact multiple of $2^k$. The construction for the more general case uses essentially the same ideas, and we omit the details in the interest of space.

Let $L = \log_2(\frac{4}{\epsilon})$ denote the number of active levels: this is the same for both $\mathcal{F}_{2^k}(2^k,\epsilon)$ and $\mathcal{F}_{2^k}(2^{k+1},\epsilon)$, since $L$ is a function of $\epsilon$ alone. We can easily verify that for $\ell < L$, the level-$\ell$ active

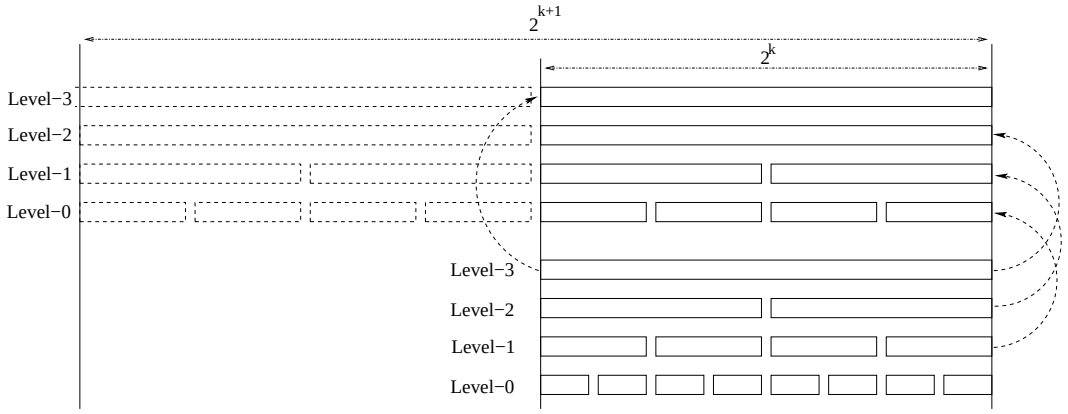


*Figure 3:* Construction of $\mathcal{F}_{2^k}(2^{k+1}, \epsilon)$ from $\mathcal{F}_{2^k}(2^k, \epsilon)$

blocks of $\mathcal{F}_{2^k}(2^{k+1}, \epsilon)$ are the same as level-$(\ell+1)$ active blocks of $\mathcal{F}_{2^k}(2^k, \epsilon)$. Let $\hat{\epsilon}$ denote the error parameter for the sketches of level-$\ell$ blocks of $\mathcal{F}_{2^k}(2^{k+1}, \epsilon)$. Then, it is easy to verify that $\mathcal{F}_{2^k}(2^k, \epsilon)$ stores $\frac{\hat{\epsilon}}{2}$-sketches of its level-$(\ell+1)$ blocks. We construct an $\hat{\epsilon}$-sketch for an active level-$\ell$ block of using the $\frac{\hat{\epsilon}}{2}$-sketch for the same block maintained by $\mathcal{F}_{2^k}(2^k, \epsilon)$. (Lemma 6.2 shows that this is possible).

There are two possible scenarios for level-$L$ blocks of $\mathcal{F}_{2^k}(2^{k+1}, \epsilon)$: there exists no active level-$L$ block for $\mathcal{F}_{2^k}(2^{k+1}, \epsilon)$, or a level-$L$ block is in under-construction state. $\mathcal{F}_{2^k}(2^{k+1}, \epsilon)$, by definition, cannot have an active level-$L$ block. For the first case, we need not do anything. To handle the second case, we make one minor assumption about the running of one-pass algorithms used to construct sketches: for any given level, we assume that we do not stop an instance of the one-pass algorithm and discard its state, until the next one for the level begins. None of the results so far depend on when we terminate the one-pass algorithms and start the next one. Under this assumption, an instance of the one-pass algorithm, capable of producing an $\epsilon_L$-sketch is running over the elements of the single active level-$L$ block of $\mathcal{F}_{2^k}(2^k, \epsilon)$. We simply continue this algorithm on behalf of the level-$L$ block of $\mathcal{F}_{2^k}(2^{k+1}, \epsilon)$ that is under construction, and after the next $2^k$ elements arrive, this algorithm can be used to produce an $\epsilon_L$-sketch. □

**Lemma 6.2** *For quantiles (resp. counts), an $\epsilon$-sketch space for a bag that uses $O(\frac{1}{\epsilon})$ can be obtained from an $\frac{\epsilon}{2}$-sketch for the bag.*

**Proof:** First consider quantiles. An $\epsilon$-sketch can be constructed by probing $\frac{\epsilon}{2}$ sketch for quantiles $\phi = \epsilon, 2\epsilon, \ldots, 1$ and storing the returned approximate quantiles. Clearly, this sketch uses $O(\frac{1}{\epsilon})$ space.

For counts, retrieve all pairs $\langle e, \tilde{f}_e \rangle$ from the $\frac{\epsilon}{2}$-sketch, and store as part of the $\epsilon$-sketch those pairs with $\tilde{f}_e > \epsilon M/2$, where $M$ is the size of the bag. At most $\frac{2}{\epsilon}$ elements can have $\tilde{f}_e > \epsilon M/2$ (recall that the approximate count $\tilde{f}_e \leq$ the frequency). Therefore, size of the constructed sketch is $O(\frac{1}{\epsilon})$. □

**Theorem 6.1** *$\epsilon$-approximate frequency counts over variable-size windows require $O(\frac{1}{\epsilon}\log^2\frac{1}{\epsilon}\log \epsilon n)$ space, where $n$ denotes the number of elements in the current window.*

**Proof:** From Lemma 6.1, Properties I and II are true for counts. Therefore, $\mathcal{V}(n, \epsilon)$ faithfully maintains $\epsilon$-approximate counts over a variable-size window having current size $n$. From Theorem 4.1, space requirements for sketch $\mathcal{F}_n(N, \epsilon)$ are at most $O(\frac{1}{\epsilon}\log^2\frac{1}{\epsilon})$, independent of $n$ and $N$. Overall, $\mathcal{V}(n, \epsilon)$ is a collection of at most $\log_2 \epsilon n$ such sketches. Therefore, total space required is $O(\frac{1}{\epsilon}\log^2\frac{1}{\epsilon}\log \epsilon n)$. □

**Theorem 6.2** *$\epsilon$-approximate quantiles over variable-size windows require $O(\frac{1}{\epsilon}\log\frac{1}{\epsilon}\log n\log\epsilon n)$ space, where $n$ denotes the number of elements in the current window.*

**Proof:** From Lemma 6.1, Properties I and II are true for quantiles. Therefore, $\mathcal{V}(n,\epsilon)$ correctly maintains $\epsilon$-approximate quantiles over a variable-size window having current size $n$. From Theorem 4.2, space requirements for sketch $\mathcal{F}_n(2^i,\epsilon)$ are at most $O(\frac{1}{\epsilon}\log\frac{1}{\epsilon}\log 2^i)$. Overall, $\mathcal{V}(n,\epsilon)$ is a collection of at most $\log_2\epsilon n$ such sketches, and the highest value of $2^i$ is less than $2n$. Therefore, total space required is $O(\frac{1}{\epsilon}\log\frac{1}{\epsilon}\log n\log\epsilon n)$. □

## 7 Randomized Algorithms for Variable-Size Windows

Our randomized algorithm for counts will use the general technique outlined in the previous section, maintaining $\mathcal{V}(n,\epsilon)$ exactly as described earlier. All that remains to be shown is that Properties I and II are satisfied by the set of triples we had maintained in Section 5, one triple per element that was sampled.

**Lemma 7.1** *Both properties are satisfied by the 1-in-$2^k$ style* STICKY SAMPLING *described in Section 5.*

**Proof:** Property I follows from the fact that the absolute error in individual counts caused by restricting the samples to the last $n$ out of $N$ elements, leads to an absolute error of at most $\epsilon N$. The relative error is at most $\frac{\epsilon N}{n}$.

Property II follows from the fact that a sample chosen uniformly out of $2^{k+1}$ successive elements is equivalent to splitting these elements into $2^k$ disjoint groups, choosing one sample from each group and finally, choosing one of these samples at random. □

**Theorem 7.1** *$\epsilon$-approximate frequency counts over variable-size sliding windows can be maintained in $O(\frac{1}{\epsilon}\log\frac{1}{\epsilon\delta}\log\epsilon n)$ space, where $\delta$ denotes the probability of failure and $n$ denotes the number of elements in the current window.*

**Proof:** The number of different sketches in $\mathcal{V}(n,\epsilon)$ is at most $\log_2\epsilon n$. From Theorem 5.1, each of these sketches requires $O(\frac{1}{\epsilon}\log\frac{1}{\epsilon\delta})$ space. Multiplying the two, we get $O(\frac{1}{\epsilon}\log\frac{1}{\epsilon\delta}\log\epsilon n)$ space. □

Finally, we present a randomized algorithm over variable-size windows for approximate quantiles.

**Theorem 7.2** *$\epsilon$-approximate quantiles over variable-size sliding windows require $O(\frac{1}{\epsilon}\log\frac{1}{\epsilon}\log^2(\frac{1}{\epsilon}\log\frac{1}{\epsilon\delta}))$ space, where $\delta$ denotes the probability of failure and $N$ denotes the number of elements in the current window.*

**Proof:** The key idea is the same as in Theorem 5.2. We replace the Greenwald-Khanna algorithm with a randomized algorithm obtained by hooking together 1-in-$2^k$ sampling with Greenwald-Khanna, splitting the error between the two steps, as shown in [MRL98]. The effect is to replace $N$ by the expression $\frac{1}{\epsilon^2}\log\frac{1}{\epsilon\delta}$ in Theorem 6.2 to arrive at the space complexity. □


## Acknowledgments

We thank Mayank Bawa for extremely useful feedback on an initial draft.

## References


[AIS93] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proc. ACM Intl. Conf. on Management of Data*, May 1993.

[BBD+02] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *Proc. of the 21st ACM Symp. on Principles of Database Systems*, pages 1–16, June 2002.

[BDM02] B. Babcock, M. Datar, and R. Motwani. Sampling from a moving window over streaming data. In *Proc. of the 13th Annual ACM-SIAM Symp. on Discrete Algorithms*, pages 633–634, Jan. 2002.

[BDMO03] B. Babcock, M. Datar, R. Motwani, and L. O'Callaghan. Maintaining variance and $k$-medians over data stream windows. In *Proc. of the 22nd ACM Symp. on Principles of Database Systems*, pages 234–243, June 2003.

[BFP+73] M. Blum, R. W. Floyd, V. R. Pratt, R. L. Rivest, and R. E. Tarjan. Time Bounds for Selection. *Journal of Computer and System Sciences*, 7:448–461, 1973.

[CCFC02] M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams. In *Proc. of the 29th Intl. Coll. on Automata, Languages, and Programming*, pages 693–703, July 2002.

[CM03] G. Cormode and S. Muthukrishnan. What's hot and what's not: tracking most frequent items dynamically. In *Proc. of the 22nd ACM Symp. on Principles of Database Systems*, pages 296–306, June 2003.

[DGIM02] M. Datar, A. Gionis, P. Indyk, and R. Motwani. Maintaining stream statistics over sliding windows. In *Proc. of the 13th Annual ACM-SIAM Symp. on Discrete Algorithms*, pages 635–644, Jan. 2002.

[DLOM02] E. D. Demaine, A. López-Ortiz, and J. I. Munro. Frequency estimation of internet packet streams with limited space. In *Proc. of the 10th Annual European Symp. on Algorithms*, pages 348–360, Sept. 2002.

[DNS91] D. DeWitt, J. Naughton, and D. Schneider. Parallel Sorting on a Shared-Nothing Architecture using Probabilistic Splitting. *Intl. Conf. on Parallel and Distributed Information Systems*, pages 280–291, 1991.

[FR75] R. W. Floyd and R. L. Rivest. Expected Time Bounds for Selection. *Communications of the ACM*, 18:165–172, 1975.

[GK01] M. Greenwald and S. Khanna. Space-efficient online computation of quantile summaries. In *Proc. of the 2001 ACM SIGMOD Intl. Conf. on Management of Data*, pages 58–66, May 2001.

[GM98] P. Gibbons and Y. Matias. New sampling-based summary statistics for improving approximate query answers. In *Proc. of the 1998 ACM SIGMOD Intl. Conf. on Management of Data*, pages 331–342, 1998.

[GT02] P. B. Gibbons and S. Tirthapura. Distributed streams algorithms for sliding windows. In *Proc. of the 14th ACM Symp. on Parallel Algorithms and Architectures*, pages 63–72, Aug. 2002.

[KSP03] R. M. Karp, S. Shenker, and C. H. Papadimitriou. A simple algorithm for finding frequent elements in streams and bags. *ACM Trans. on Database Systems*, 28(1):51–55, Mar. 2003.

[LLXY04] X. Lin, H. Lu, J. Xu, and J. X. Yu. Continuously maintaining quantile summaries of the most recent N elements over a data stream. In *Proc. of the 20th Intl. Conf. on Data Engineering*, Mar. 2004. To appear.

[MG82] J. Misra and D. Gries. Finding repeated elements. *Science of Computer Programming*, 2(2):143–152, Nov. 1982.

[MM02] G. S. Manku and R. Motwani. Approximate frequency counts over data streams. In *Proc. of the 28th Intl. Conf. on Very Large Data Bases*, pages 356–357, Aug. 2002.

[MP80] J. I. Munro and M. Patterson. Selection and sorting with limited storage. *Theoretical Computer Science*, 12:315–323, 1980.

[MRL98] G. S. Manku, S. Rajagopalan, and B. G. Lindsay. Approximate medians and other quantiles in one pass and with limited memory. In *Proc. of the 1998 ACM SIGMOD Intl. Conf. on Management of Data*, pages 426–435, June 1998.

[MRL99] G. S. Manku, S. Rajagopalan, and B. G. Lindsay. Random sampling techniques for space efficient online computation of order statistics of large datasets. In *Proc. of the 1999 ACM SIGMOD Intl. Conf. on Management of Data*, pages 251–262, June 1999.

[Pat96] M. R. Paterson. Progress in selection. In *Proc. 5th Scandinavian Workshop on Algorithm Theory*, pages 368–379, 1996.

[Poh69] I. Pohl. A Minimum Storage Algorithm for Computing the Median. Technical Report IBM Research Report RC 2701 (# 12713), IBM T J Watson Center, Nov. 1969.

[Toi96] H. Toivonen. Sampling large database for association rules. In *Proc. of the 22nd Intl. Conf. on Very Large Data Bases*, pages 134–145, 1996.

[Vit85] J. Vitter. Random sampling with a reservoir. *ACM Trans. on Mathematical Software*, 11(1):37–57, Mar. 1985.

# Appendix

We present an alternative proof for our algorithms over variable-size windows. Our exposition gives more insight into the relationships between various blocks and sketches that we maintain.

## A Deterministic Algorithms for Variable-Size Windows

We first define blocks and layers over a data stream. Layers are numbered sequentially as $0, 1, 2, \ldots$. Each layer is composed of equi-sized pair-wise disjoint blocks such that the union of all blocks covers the entire stream. For $\ell \geq 1$, each block in layer $\ell$ is the union of two adjacent blocks in layer $\ell - 1$. For $\ell \geq 0$, all blocks in layer $\ell$ have size $2^\ell$. Our definitions of blocks and layers are identical to those in Section 4 – the only difference is the block-size in layer 0, which has been set to 1 for variable-size windows.

A block in layer $\ell \geq 0$ is said to be *active* if all of its members are among the last $\min(N, 4 \cdot 2^\ell/\epsilon)$ positions from now. Thus there are at most $4/\epsilon$ active blocks in layer $\ell \geq 0$, each of size $2^\ell$. It follows that the total number of active blocks is $O(\frac{1}{\epsilon} \log \epsilon N)$. A block is said to be *under construction* if some of its members have already arrived but at least one member is yet to arrive. The total number of blocks under construction is at most $\log_2 N$.

Let $L = \log_2 \frac{4}{\epsilon}$. For $0 \leq i \leq L$, we define error parameter $\epsilon_i = \frac{2^{-i}}{L+1}$. For fixed-size windows, we maintained exactly one sketch per active block. However, for variable-size windows, multiple sketches with various error parameters are associated with an active block, as per the following two rules:

(a) For every active block, irrespective of the layer it belongs to, there is a sketch with parameter $\epsilon_0$.

(b) For $1 \leq i \leq L$, a sketch for an active block (in layer $\ell$) with error parameter $\epsilon_i$ is remembered only if sketches with error parameter $\epsilon_{i-1}$ are being remembered for both of its sub-blocks (in layer $\ell - 1$).

For all blocks under construction at level $i$ or more, we run a single-pass algorithm (Misra-Gries for counts, Greenwald-Khanna for quantiles) with parameter $\epsilon_i$. When the state of a block changes from under construction to active, the sketch for the block is obtained by probing the data-structure maintained by the one-pass algorithm (see Section 4 for more details).

*Space requirements for active blocks:* For $0 \leq i \leq L$, let $B_i$ denote the total number of blocks which maintain sketches with error parameter $\epsilon_i$. From Rule (b) above, it follows that $B_i \leq 2^{-i} B_0$. From the definition of error parameters, $\epsilon_i = 2^{-i}\epsilon_0$. Since the space requirement for one active block with error parameter $\epsilon_i$ is at most $1/\epsilon_i$ (for both counts and quantiles), the total space requirements are $\frac{B_0}{\epsilon_0} + \frac{B_1}{\epsilon_1} + \cdots + \frac{B_L}{\epsilon_L} \leq (L+1)\frac{B_0}{\epsilon_0}$. $B_0$ equals the total number of active blocks, which is $O(\frac{1}{\epsilon} \log \epsilon N)$. Error parameter $\epsilon_0 = \frac{1}{L+1}$ and $L = \log \frac{4}{\epsilon}$. Therefore, $(L+1)\frac{B_0}{\epsilon_0} = O(\frac{1}{\epsilon} \log^2 \frac{1}{\epsilon} \log \epsilon N)$.

*Space requirements for blocks under construction:* At any time, there are at most $\log_2 N$ blocks (one per layer) under construction. For each block at level $\ell > 0$, there are multiple sketches under construction. For $0 \leq i \leq L$, all blocks above and including level $i$ have a sketch under construction with error parameter $\epsilon_i$. For Misra-Gries algorithm, a sketch under construction with parameter $\epsilon_i$ requires $O(1/\epsilon_i)$ space. Therefore, blocks under construction taken together require no more than $O(\frac{1}{\epsilon} \log \frac{1}{\epsilon} \log N)$ space. For Greenwald-Khanna algorithm, the space requirements are $O(\frac{1}{\epsilon_i} \log \epsilon_i N)$, since a block cannot have more than $N$ elements in it. Therefore, blocks under construction taken together require no more than $O(\frac{1}{\epsilon} \log^2 \frac{1}{\epsilon} \log N)$ space.

*Retrieval of approximate counts and quantiles by merging sketches:* We will show that it is possible to retrieve $\epsilon$-approximate quantiles for any $n \leq N$, where $N$ is the current size of the window. Consider layer $k$ where $2^{k-1} < n \leq 2^k$. Each block at layer $k$ is comprised of $2^k$ elements. Consider the following sketches at $L+1$ different layers: all sketches with error parameter $\epsilon_0$ at layer $k$, all sketches with error parameter $\epsilon_1$ at layer $k+1$, all sketches with error parameter $\epsilon_2$ at layer $k+2$, and so on. The union of

these sketches is exactly the set of sketches we would have maintained had we used our earlier algorithm (see Section 4) for a *fixed-size* window having size $2^k$. In Lemma 6.1, we saw that Property I is true for such a sketch, allowing us to retrieve $\frac{\epsilon}{2}$-approximate quantiles (or counts) for any window whose size is at least $2^{k-1}$.

The above discussion yields the following theorems.

**Theorem A.1** *$\epsilon$-approximate frequency counts over variable-size windows require $O(\frac{1}{\epsilon}\log^2\frac{1}{\epsilon}\log\epsilon N)$ space, where $N$ denotes the number of elements in the current window.*

**Theorem A.2** *$\epsilon$-approximate quantiles over variable-size windows require $O(\frac{1}{\epsilon}\log^2\frac{1}{\epsilon}\log\epsilon N)$ space, where $N$ denotes the number of elements in the current window.*

## B Randomized Algorithms for Variable-Size Windows

We now present a variation on STICKY SAMPLING that allows us to maintain approximate frequency counts over variable-size windows. We use the same definition of layers and blocks as in the previous section. A block at layer $\ell$ is comprised of $2^\ell$ elements. The definition of active blocks is different: A block in layer $\ell$ is said to be active if all its elements are within the last $\min\{N, \frac{2^\ell}{\epsilon}\log\frac{1}{\epsilon\delta}\}$ positions in the stream. It follows that the number of active blocks is $O(\frac{1}{\epsilon}\log\frac{1}{\epsilon\delta}\log\epsilon N)$. For an active block at layer $\ell \geq 0$, we sample exactly 1-in-$2^\ell$ elements in the block. For each sample, we maintain a triple $\langle$element, count, position$\rangle$, where count denotes the number of occurrences of this element before it is sampled again at this level and position denotes the position within the stream at which this triple was created.

**Theorem B.1** *$\epsilon$-approximate frequency counts over variable-size sliding windows can be maintained in $O(\frac{1}{\epsilon}\log\frac{1}{\epsilon\delta}\log\epsilon N)$ space, where $\delta$ denotes the probability of failure and $N$ denotes the number of elements in the current window.*

**Proof:** The space requirement equals the number of active blocks, which is $O(\frac{1}{\epsilon}\log\frac{1}{\epsilon\delta}\log\epsilon N)$. We now show how $\epsilon$-approximate counts can be derived from the samples for any $n \leq N$. Let $k$ be such that $2^{k-1} < n/(\frac{1}{\epsilon}\log\frac{1}{\epsilon\delta}) \leq 2^k$. The set of triples in layer $k$ is exactly those maintained using the algorithm for a *fixed-size* sliding window of size $2^k$. We can construct $\epsilon$-approximate counts from these samples. □