Maintaining Bernoulli Samples over Evolving Multisets

Rainer Gemulla Wolfgang Lehner
Technische Universitat Dresden
01099 Dresden, Germany
{gemulla,lehner}@inf.tu-dresden.de

Peter J. Haas
IBM Almaden Research Center
San Jose, California, USA
phaas@us.ibm.com

ABSTRACT

Random sampling has become a crucial component of modern data management systems. Although the literature on database sampling is large, there has been relatively little work on the problem of maintaining a sample in the presence of arbitrary insertions and deletions to the underlying dataset. Most existing maintenance techniques apply either to the insert-only case or to datasets that do not contain duplicates. In this paper, we provide a scheme that maintains a Bernoulli sample of an underlying multiset in the presence of an arbitrary stream of updates, deletions, and insertions. Importantly, the scheme never needs to access the underlying multiset. Such Bernoulli samples are easy to manipulate, and are well suited to parallel processing environments. Our method can be viewed as an enhancement of the "counting sample" scheme developed by Gibbons and Matias for estimating the frequency of highly frequent items. We show how the "tracking counters" used by our maintenance scheme can be exploited to estimate population frequencies, sums, and averages in an unbiased manner, with lower variance than the usual estimators based on a Bernoulli sample. The number of distinct items in the multiset can also be estimated without bias. Finally, we discuss certain problems of subsampling and merging that arise in systems with limited memory resources or distributed processing, respectively.

Categories and Subject Descriptors

H.2 [**Database Management**]: Miscellaneous; G.3 [**Probability and Statistics**]: Probabilistic algorithms

General Terms

Algorithms, theory

Keywords

Bernoulli multiset sampling, incremental sample maintenance

1. INTRODUCTION

Random sampling is an essential component of modern data management systems, with applications including approximate query answering, query optimization, summarization of data streams, network monitoring, and data mining. The traditional literature on sampling from databases—see, for example, [12]—typically views the database as static, and focuses on the problem of producing and exploiting a one-time sample in an efficient manner. However, as discussed in [7], even the best algorithms for materializing a sample on the fly, in response to a user request, can be too expensive. An appealing alternative approach is to materialize a sample and incrementally maintain it as the underlying dataset evolves; thus a sample is always available when needed.

For this latter approach to be successful, it is important to avoid accesses to the underlying dataset, and touch only the sample itself, together with the stream of update, delete, and insert (UDI) transactions to the dataset. The reason for this is that database accesses are typically much more expensive than accesses to the sample. For instance, database accesses usually require I/O operations on disks, whereas samples are often stored in main memory. In distributed systems, the underlying dataset may reside at a remote location. Experiments in [7] show that algorithms requiring access to the underlying dataset are not competitive in terms of cost. Moreover, in many streaming data systems, the underlying dataset may not be accessible at all. We therefore restrict attention to maintenance algorithms that do not touch the underlying dataset.

We follow the literature in concentrating on the maintenance of uniform samples, because such samples are popular, flexible, and form the basis for more complex sampling schemes. The majority of known techniques for maintaining uniform samples apply either to the insert-only case [2, 11, 14] or to datasets that do not contain duplicates [1, 7], i.e., datasets that are true sets, rather than the multisets that commonly appear in relational and other systems. In contrast, the work in [10] on "counting samples" handles arbitrary UDI transactions on multisets, using subsampling techniques to ensure that the sample footprint stays within a specified upper bound. The counting sample is also used to estimate answers to "hot list" queries, i.e., to estimate frequencies (in the dataset) of highly frequent items. As shown in [2], the use of subsampling actually causes the samples to be non-uniform, although the degree of non-uniformity may be acceptable in practice.

In this paper, we complement the results in [10] by providing an algorithm for incrementally maintaining a Bernoulli sample over an evolving multiset (in which a given item may be duplicated one or more times). The algorithm can handle arbitrary UDI transactions and does not require any accesses of the underlying data. Although Bernoulli samples do not have a bounded footprint, they are truly uniform (see Section 2.1). Moreover, Bernoulli samples are easy to work with and are well-suited for parallel processing. Specifically, they can be maintained, independently and in parallel, for each partition of a distributed dataset, and the samples can be merged at any

PODS'07, June 11–14, 2007, Beijing, China.

time to obtain a Bernoulli sample of the entire dataset. Bernoulli samples can also be easily subsampled, in order to reduce their space requirements.[1]

In order to maintain a Bernoulli sample under UDI transactions, our method augments the sample with "tracking counters," originally introduced in [10] for the purpose of estimating the population frequencies of "hot" items. We show that these counters not only facilitate maintenance, but can also exploited to obtain unbiased estimators of population frequencies, sums, and averages, where these estimators have lower variance than the usual estimators based on an ordinary Bernoulli sample. We also show how to estimate the number of distinct items in the multiset in an unbiased manner. We derive the standard error for each of our estimators, and provide formulas for estimating these standard errors.

Finally, we examine issues that arise when incrementally maintaining samples that result from subsampling or merging operations. We show how to subsample one of our new augmented samples to obtain a new, smaller augmented sample that can be incrementally maintained. As a negative result, we show that, in the general case of non-disjoint parent datasets, it is impossible to merge augmented samples to obtain a new augmented sample from the union of the corresponding parent datasets.

2. PRELIMINARIES

We first discuss classical Bernoulli sampling and maintenance schemes over ordinary sets, and then give a naive maintenance scheme that handles multisets and UDI transactions. Throughout the paper, we view an update to a dataset as a deletion followed by an insertion, and therefore do not discuss updates separately.

We consider a (possibly infinite) set $T = \{t_1, t_2, \ldots\}$ of unique, distinguishable items that are inserted into and deleted from a dataset R over time. For example, T might correspond to a finite set of IP addresses or social security numbers, or the infinite set of nonnegative integers (perhaps representing order sizes). Denote by $\gamma = (\gamma_1, \gamma_2, \ldots)$ an infinite sequence of transactions, where each transaction γ_i is either of the form $+t_k$, which corresponds to the insertion of item t_k into R, or of the form $-t_k$, which corresponds to the deletion of item t_k from R. In general, items that are deleted may be subsequently reinserted. We restrict attention to "feasible" sequences such that $\gamma_i = -t_k$ only if item t_k is in the dataset just prior to the processing of the ith transaction. In principle, when R is a multiset, $m > 1$ copies of an item t_k may be inserted into R or deleted from R in the course of a transaction; we include such a compound transaction within our framework by viewing it as a sequence of m simple transactions, each involving a single insertion or deletion.

We denote by R_i and S_i the state of the dataset and sample, respectively, just after processing transaction γ_i. Without loss of generality, we assume throughout that the initial state of the dataset and sample are given by $R_0 = S_0 = \emptyset$.

2.1 Bernoulli Sampling on Sets

In classical Bernoulli sampling, the transaction sequence γ consists of insertions only, and there are no duplicates, so that an item in T is inserted at most once, and R is a ordinary set. For a specified sampling rate $q \in [0,1]$, each inserted item is included in the sample S with probability q and excluded with probability $1-q$, independent of the other items. We refer to this scheme as BERN(q) sampling.

[1] See [2] and the discussion in Section 2.1 for further details on manipulation of Bernoulli samples and their use in a "synopsis warehouse" architecture.

This sampling scheme is *uniform* in that it produces any two samples of the same size with equal probability. Other uniform sampling schemes are simple random sampling with and without replacement. Whereas the size of a simple random sample is fixed, the size of a Bernoulli sample is random, having a binomial distribution. Specifically, after processing i insertion transactions using BERN(q) sampling, we have $P(|S_i| = k) = B(k; i, q)$ for $0 \le n \le m$, where we use the notation

$$B(n; m, q) = \binom{m}{n} q^n (1-q)^{m-n}$$

to denote a binomial probability, with $B(n; m, q) = 0$ when $m < n$ or $n < 0$.

Bernoulli samples are easy to manipulate. Because items are included or excluded independently of each other, it follows immediately that, if S_1 and S_2 are BERN(q) samples of R_1 and R_2, respectively, and if $R_1 \cap R_2 = \emptyset$, then $S_1 \cup S_2$ is a BERN(q) sample of $R_1 \cup R_2$. Moreover, if S is a BERN(q) sample of R and S' is a BERN(q') sample of S, then S' is a BERN($q'q$) sample of R. Indeed, an item appears in the subsample S' if and only if (1) it appears in the initial sample S and (2) it is retained during subsampling. The probability of these events is q and q', respectively, and these events are independent, so that the ultimate probability of including an item in the subsample is $q'q$. Items are included or excluded independently of each other during this two-phase process, so that the process coincides with BERN($q'q$) sampling from R.

It is straightforward to extend Bernoulli sampling on sets to handle deletions. For an insertion transaction $\gamma_i = +t_k$, simply set $S_i = S_{i-1} \cup \{t_k\}$ with probability q and $S_i = S_{i-1}$ with probability $1-q$. For a deletion transaction $\gamma_i = -t_k$, set $S_i = S_{i-1} \setminus \{t_k\}$, i.e., remove item t_k from the sample if it is present. Thus the deletion operation "annihilates" item t_k; it is as if item t_k were never inserted into R. This scheme is called MBERN(q) sampling in [7]. Observe that this procedure never accesses the underlying dataset R, so that sampling and sample maintenance trivially coincide.

We can now anticipate why maintaining a Bernoulli sample of a multiset is harder than maintaining a Bernoulli sample of an ordinary set. When processing a deletion transaction $\gamma_i = -t_k$, there may be multiple copies of item t_k, both in R and in S, so it is not immediately clear how to proceed.

2.2 Bernoulli Sampling on Multisets

Now suppose that an item $t \in T$ can appear more than once in the dataset, and hence in the sample. Then both the dataset and the sample are multisets. Denote by $X_i(t)$ the frequency of item $t \in T$ in the sample S_i and by $N_i(t)$ the frequency of item t in the dataset R_i. Note that $N_i(t)$ is completely determined by the sequence γ, whereas $X_i(t)$ is a random variable. In this setting, S_i is a BERN(q) sample of R_i if and only if

$$P(X_i(t) = k_t \text{ for } t \in T) = \prod_{t \in T} B(k_t; N_i(t), q)$$

for any set of nonnegative integers $\{k_t : t \in T\}$. That is, each $X_i(t)$ is binomially distributed, and the random variables $\{X_i(t) : t \in T\}$ are mutually independent.

We assume that the sample is physically stored in a compressed representation [10]. In this representation, each element of the sample comprises a pair $(t, X_i(t))$ when $X_i(t) > 1$ or a singleton (t) when $X_i(t) = 1$. An item t with $X_i(t) = 0$ does not appear in the sample.

Notice that, in any Bernoulli sample of a multiset, each item is maintained independently of the other items. That is, the value of $X_i(t)$ remains unaffected if an item $t' \neq t$ is inserted or removed.

Without loss of generality, therefore, we henceforth fix an item t and focus on the maintenance of $X_i(t)$ as the transaction sequence γ is processed. We therefore assume that γ consists solely of insertions and deletions of item t.[2] For brevity, we set $X_i = X_i(t)$ and $N_i = N_i(t)$. For $i = 0$, the sample is empty and we have $X_i = 0$.

With these definitions in hand, we now describe a naive algorithm for maintaining a Bernoulli sample of a multiset in the presence of UDI transactions. We compute S_{i+1} from S_i and γ_{i+1} as follows. If $\gamma_{i+1} = +t$ (an insertion), then

$$X_{i+1} = \begin{cases} X_i + 1 & \text{with probability } q \\ X_i & \text{with probability } 1-q. \end{cases} \quad (1)$$

If $\gamma_{i+1} = -t$ (a deletion), then

$$X_{i+1} = \begin{cases} X_i - 1 & \text{with probability } X_i/N_i \\ X_i & \text{with probability } 1-(X_i/N_i) \end{cases} \quad (2)$$

The following theorem asserts the correctness of the naive algorithm.

THEOREM 1. *For any non-negative integer $k \leq N_i$, the naive algorithm maintains the invariant*

$$P(X_i = k) = B(k; N_i, q). \quad (3)$$

PROOF. The proof is by induction on i. For $i = 0$, the equality in (3) holds trivially. Now suppose that (3) holds for some i. If $\gamma_{i+1} = +t$, then $N_{i+1} = N_i + 1$ and, using the induction hypothesis and some straightforward algebra,

$$\begin{aligned} P(X_{i+1} = k) &= (1-q)P(X_i = k) + qP(X_i = k-1) \\ &= (1-q)B(k; N_i, q) + qB(k-1; N_i, q) \\ &= B(k; N_{i+1}, q). \end{aligned}$$

Similarly, if $\gamma_{i+1} = -t$, then $N_{i+1} = N_i - 1$ and

$$\begin{aligned} &P(X_{i+1} = k) \\ &= \frac{k+1}{N_i} P(X_i = k+1) + \frac{N_i - k}{N_i} P(X_i = k) \\ &= \frac{k+1}{N_i} B(k+1; N_i, q) + \frac{N_i - k}{N_i} B(k; N_i, q) \\ &= B(k; N_{i+1}, q). \end{aligned}$$

and the desired result follows. □

A less formal but more intuitive probabilistic argument is as follows. Conceptually, the naive algorithm uses a "random pairing" approach, in which each deletion transaction is paired with (and hence annihilates) a previous insertion transaction, chosen randomly and uniformly from all such (currently non-annihilated) insertion transactions. If the deletion γ_i is paired with a previous insertion γ_{i-j} and if transaction γ_{i-j} had originally resulted in a sample inclusion, then the annihilated item is removed from the sample as well as from the dataset. The probability that a deletion transaction is paired with an insertion transaction that corresponds to an item in the sample is X_i/N_i.

Unfortunately, the naive algorithm is unusable, because processing a deletion requires knowledge of the quantity N_i, which has to be obtained from the underlying dataset R_i. As discussed previously, R_i is usually expensive, and sometimes impossible, to access. One might consider maintaining the counters N_i locally for each distinct item in the dataset, but this is equivalent to storing the dataset itself, which is typically infeasible in practice.

[2]The exception to this convention is Section 7, where we give the complete pseudocode for our algorithms.

3. A NOVEL APPROACH

As indicated in the previous section, in order to support Bernoulli sample maintenance in the presence of general UDI transactions to a multiset, one would like somehow to maintain only the $N_i(t)$ counters corresponding to items t that are in the sample. Such maintenance is impossible, because insertions into the dataset of an item that is not currently in the sample, but will eventually be in the sample, cannot be properly accounted for. Borrowing an idea from [10], our new maintenance method rests on the fact that it suffices to maintain a "tracking counter" $Y_i(t)$ for each item t in the sample. Whenever $X_i(t)$, the frequency of t in the sample, is positive, the counter $Y_i(t)$ records the number of (non-annihilated) insertions of t into the dataset since the first of the current $X_i(t)$ sample items was inserted; note that the dataset insertion corresponding to the first of these $X_i(t)$ sample inclusions is counted as part of $Y_i(t)$. We therefore modify the general layout of the sample S_i as follows: for each distinct item $t \in T$ that occurs in the sample at least once, S_i contains the triple $(t, X_i(t), Y_i(t))$. To save space, we store the entry for t as $(t, X_i(t), Y_i(t))$ if $Y_i(t) > 1$ and simply as (t) if $X_i(t) = Y_i(t) = 1$. The resulting space savings can be significant when there are many unique values in the dataset.

3.1 Algorithmic Description

As in previous sections, we focus on the case of a single item t, and represent the state of S_i as (X_i, Y_i); i.e., we suppress the dependence on t in our notation. We also set $X_i = Y_i = 0$ whenever $t \notin S_i$. For $i = 0$, the sample is empty and we have $X_i = Y_i = 0$.

The new algorithm works as follows: If $\gamma_{i+1} = +t$, then

$$(X_{i+1}, Y_{i+1}) = \begin{cases} (X_i + 1, Y_i + 1) & \text{if } \Phi_{i+1} = 1 \\ (X_i, Y_i + 1) & \text{if } \Phi_{i+1} = 0, X_i > 0 \\ (0,0) & \text{otherwise,} \end{cases} \quad (4)$$

where Φ_{i+1} is a 0/1 random variable such that $P(\Phi_{i+1} = 1) = q$. If $\gamma_{i+1} = -t$, then

$$(X_{i+1}, Y_{i+1}) = \begin{cases} (0,0) & \text{if } X_i = 0 \\ (0,0) & \text{if } X_i = Y_i = 1 \\ (X_i - 1, Y_i - 1) & \text{if } X_i \geq 1, Y_i > 1, \Psi_{i+1} = 1 \\ (X_i, Y_i - 1) & \text{otherwise,} \end{cases} \quad (5)$$

where Ψ_{i+1} is a 0/1 random variable such that

$$P(\Psi_{i+1} = 1) = \frac{X_i - 1}{Y_i - 1}.$$

Note that item t is removed from the sample if $X_i > 0, X_{i+1} = 0$ and added to the sample if $X_i = 0, X_{i+1} > 0$. The processing of γ_{i+1} is solely based on S_i and γ_{i+1}, that is, access to the dataset R is not required at any time.

3.2 An Example

Figure 1 depicts a probability tree for our algorithm with $\gamma = (+t, +t, +t, -t)$ and $q = 0.25$. Each node of the tree represents a possible state of the sample; for example, $(1,2)$ stands for $S = \{(t,1,2)\}$. Edges represent state transitions and are weighted by the respective transition probability. To determine the probability of reaching a given node, multiply the probabilities along the path from the root to the node. To compute the probability that the sample is in a specified state after γ has been processed, sum the probabilities of all leaf nodes that correspond to the state. For example,

$$P(S_4 = \emptyset) = \left(\frac{3}{4}\right)^2 \frac{1}{4} + \left(\frac{3}{4}\right)^3.$$

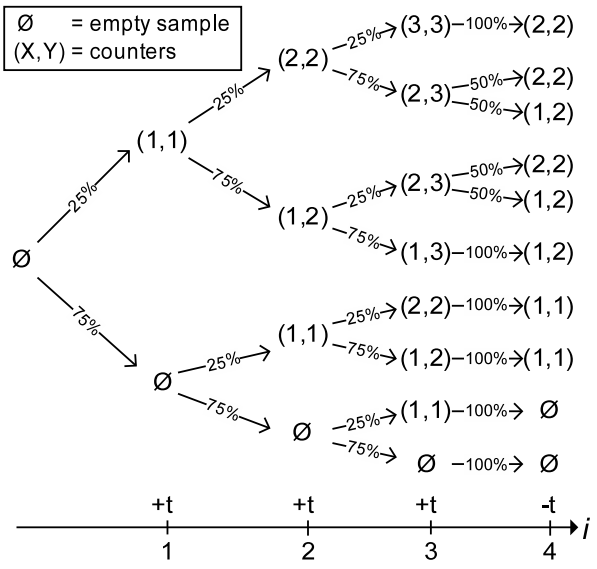

Figure 1: Example with $q = 25\%$

Summing up all such probabilities, we find that

$$P(X_4 = 0) = \frac{9}{16} = (1-q)^2$$
$$P(X_4 = 1) = \frac{6}{16} = 2q(1-q)$$
$$P(X_4 = 2) = \frac{1}{16} = q^2.$$

Thus X_4 is binomially distributed, and we have a Bernoulli sample of R_4.

3.3 Correctness of the Algorithm

To show that the algorithm of Section 3.1 indeed maintains a true Bernoulli sample, we first derive the probability distribution of the tracking counter Y_i. Recall that $N_i = |R_i|$ denotes the number of non-annihilated insertion transactions after processing the sequence $\gamma_1, \gamma_2, \ldots, \gamma_i$.

LEMMA 1. *For $i \geq 0$ and any integer $0 \leq k \leq N_i$,*

$$P(Y_i = m) = \begin{cases} (1-q)^{N_i} & \text{if } m = 0 \\ q(1-q)^{N_i - m} & \text{otherwise.} \end{cases} \tag{6}$$

PROOF. Our proof is by induction on i. We have $Y_i = N_i = 0$ when $i = 0$, and (6) holds trivially. Suppose for induction that (6) holds for i. If transaction γ_{i+1} is an insertion, then $N_{i+1} = N_i + 1$ and

$$P(Y_{i+1} = m) = \begin{cases} (1-q)P(Y_i = 0) & \text{if } m = 0 \\ qP(Y_i = 0) & \text{if } m = 1 \\ P(Y_i = m-1) & \text{otherwise.} \end{cases}$$

If transaction γ_{i+1} is a deletion, then $N_{i+1} = N_i - 1$ and

$$P(Y_{i+1} = m) = \begin{cases} P(Y_i = 0) + P(Y_i = 1) & \text{if } m = 0 \\ P(Y_i = m+1) & \text{otherwise.} \end{cases}$$

The identity in (6) now follows by applying induction hypothesis, together with some straightforward algebra. □

The assertion of Lemma 1 is particularly easy to understand in the special case where all transactions are insertions, so that there are no annihilated transactions and $N_i = i$. In this scenario, $Y_i = 0$ if all i inserted items are excluded from the sample, which occurs with probability $(1-q)^i$, and $Y_i = m > 0$ if the first $i - m$ items are excluded and the next item is included, which occurs with probability $q(1-q)^{i-m}$.

We now establish the correctness of our sample-maintenance algorithm.

THEOREM 2. *For $i \geq 0$ and any integer $0 \leq k \leq N_i$, the algorithm of Section 3.1 maintains the invariant*

$$P(X_i = k) = B(k; N_i, q). \tag{7}$$

PROOF. Although we could prove the validity of (7) by enhancing the inductive proof of Lemma 1, we choose to give an intuitive probabilistic argument that provides insight into why the algorithm works. Fix $i \geq 0$ and observe that $X_i = 0$ if and only if $Y_i = 0$, so that, using (6), we have

$$P(X_i = 0) = P(Y_i = 0) = B(0; N_i, q),$$

as asserted in (7).

Now consider a scenario in which $X_i \geq 1$, and denote by i^* the index of the transaction at which the first of the X_i instances of item t was inserted into the sample. It follows from the definition of i^* that, for $j = i^*+1, i^*+2, \ldots, i$, the cumulative number of insertions in the sequence $\gamma_{i^*,j} = (\gamma_{i^*+1}, \gamma_{i^*+2}, \ldots, \gamma_j)$ is at least as great as the cumulative number of deletions, so that the net number of insertions is always nonnegative. We can therefore view each deletion in $\gamma_{i^*,i}$ as annihilating a previous insertion that is also an element of $\gamma_{i^*,i}$, and the net effect of processing the transactions in $\gamma_{i^*,i}$ can be viewed as inserting the set of non-annihilated "new" items into the dataset R_{i^*}. For $j = i^*+1, i^*+2, \ldots, i$, denote by R'_j and S'_j the set of non-annihilated new items in the dataset and sample, respectively, just after processing transaction γ_j, and set $N'_j = |R'_j|$ and $X'_j = |S'_j|$. Observe that $X'_j = X_j - 1 \geq 0$, since the sample S_j consists of the single "old" item that was included at transaction γ_{i^*}, together with the $X_j - 1$ new items in S'_j. Also observe that $N'_j = Y_j - 1$, since Y_j counts the item inserted into the dataset at γ_{i^*} together with the non-annihilated new items inserted into the dataset while processing the transaction sequence $\gamma_{i^*,j}$.

We claim that X'_i can be viewed as the size of a sample S'_i that is obtained from R'_i by processing the transactions in $\gamma_{i^*,i}$ using the naive algorithm of Section 2.2. Indeed, for $j = i^*, i^*+1, \ldots, i-1$, suppose that transaction γ_{j+1} is an insertion. It follows from (4) that X_j, and hence X'_j, is incremented if and only if the random variable Φ_{j+1} equals 1, i.e.,

$$X'_{j+1} = \begin{cases} X'_j + 1 & \text{with probability } q \\ X'_j & \text{with probability } 1-q. \end{cases} \tag{8}$$

If γ_{j+1} is a deletion, then, by (5),

$$X'_{j+1} = \begin{cases} X'_j - 1 & \text{if } \Psi_{i+1} = 1 \\ X'_j & \text{otherwise,} \end{cases} \tag{9}$$

where

$$P(\Psi_{j+1} = 1) = \frac{X_j - 1}{Y_j - 1} = \frac{X'_j}{N'_j}.$$

Our claim follows upon comparison of (8) to (1) and of (9) to (2).

The foregoing claim and Theorem 1 together imply that S'_i is a true Bernoulli sample of R'_i. We therefore have

$$\begin{aligned} P(X_i = k \mid Y_i = m) &= P(X'_i = k-1 \mid N'_i = m-1) \\ &= B(k-1; m-1, q) \end{aligned} \tag{10}$$

for $1 \le k \le m \le N_i$. Combining (6) and (10), we find that

$$\begin{aligned} P(X_i = k) &= \sum_{m=k}^{N_i} P(X_i = k \mid Y_i = m) P(Y_i = m) \\ &= \sum_{m=k}^{N_i} B(k-1; m-1, q) q (1-q)^{N_i - m} \\ &= q^k (1-q)^{N_i - k} \sum_{m=k}^{N_i} \binom{m-1}{k-1} \\ &= B(k; N_i, q) \end{aligned}$$

for $k \ge 1$, and the desired result follows. □

Thus, the new algorithm works by tracking the net number of insertions into the dataset only after the item is first inserted into the sample, i.e., from transaction-step i^* onwards. As mentioned above, this idea originally appeared as part of the counting-sample method in [10], although our use of the idea is slightly different.

4. ESTIMATION

In this section, we derive unbiased estimators for item frequencies in R, as well as for sums and averages over R of functions of the items, and for the number of distinct items in R. We show how to exploit the tracking counters to obtain a variance reduction.

4.1 Estimating Frequencies

We again assume a single item t. Ignoring the tracking counters in our augmented Bernoulli sample, the frequency N_i of an item t can be estimated as $\hat{N}_{X_i} = X_i/q$. This estimator is the standard *Horvitz-Thompson* (HT) estimator for a Bernoulli sample from R; see [13]. The HT estimator $\hat{N}_{X_i}$ is unbiased and has variance given by

$$V[\hat{N}_{X_i}] = (1-q)N_i/q.$$

To motivate our improved estimator, write

$$\hat{N}_{X_i} = \frac{X_i - 1}{q} + \frac{1}{q}. \tag{11}$$

Recall the definitions of i^* and R_i' from the proof of Theorem 2. As described in the proof of that theorem, $X_i - 1$ is the size of a BERN(q) sample from R_i', so that the first term in (11) is simply the (unbiased) HT estimator of the frequency of t in R_i'. From (6), it follows that the number $L_i = N_i - Y_i$ of (non-annihilated) items inserted into R during the processing of $\gamma_1, \gamma_2, \ldots, \gamma_{i^*}$ has a geometric distribution:

$$P(L_i = l) = q(1-q)^{l-1} \tag{12}$$

for $l \ge 1$. The expected number of items is therefore $E[L_i] = 1/q$. Thus the second term in (11) is an estimator of the number of items in $R_{i^*} = R_i \setminus R_i'$. Since we have maintained an augmented sample $S_i = \{(X_i, Y_i)\}$, however, we know the value of the number of items inserted into R_i' *exactly*: this number is simply $Y_i - 1$. Thus, intuitively, we can reduce the variance of the estimator $\hat{N}_{X_i}$ by replacing the first term in (11) by the quantity that it is trying to estimate, yielding the improved estimator $Y_i - 1 + (1/q)$. This estimator is not quite correct, however, because there is a positive probability that $Y_i = 0$, in which case the above reasoning does not hold. When $Y_i = 0$, item t is not in the sample, and we have no information at all about N_i. The simplest choice in this case is to estimate N_i as 0, just as in the HT estimator; we show below that this choice ensures unbiasedness. Thus the final form of our improved estimator is

$$\hat{N}_{Y_i} = \begin{cases} 0 & \text{if } Y_i = 0 \\ Y_i - 1 + (1/q) & \text{otherwise} \end{cases}$$

The following result shows that $\hat{N}_{Y_i}$ is indeed unbiased.

THEOREM 3. *$E[\hat{N}_{Y_i}] = N_i$ for $i \ge 0$.*

PROOF. Fixing t and suppressing the subscript i in our notation, we have

$$\begin{aligned} E[\hat{N}_Y] &= P(Y=0) E[\hat{N}_Y \mid Y = 0] \\ &\quad + P(Y > 0) E[\hat{N}_Y \mid Y > 0] \\ &= P(Y > 0) E[Y \mid Y > 0] + P(Y > 0) \frac{1-q}{q} \end{aligned} \tag{13}$$

Thus, we have to compute $P(Y > 0)$ and $E[Y \mid Y > 0]$. From (6), the former quantity is given by

$$P(Y > 0) = 1 - P(Y = 0) = 1 - (1-q)^N. \tag{14}$$

and the latter quantity by

$$\begin{aligned} E[Y \mid Y > 0] &= \sum_{m=1}^{N} m P(Y = m \mid Y > 0) \\ &= \sum_{m=1}^{N} m \frac{P(Y=m)}{P(Y>0)} \\ &= \sum_{m=1}^{N} \frac{mq(1-q)^{N-m}}{1-(1-q)^N} \\ &= \frac{Nq - (1-q)\left(1-(1-q)^N\right)}{q\left(1-(1-q)^N\right)} \end{aligned} \tag{15}$$

The assertion follows after substituting (14) and (15) into (13). □

Calculations similar to the proof of Theorem 3 show that the variance of $\hat{N}_Y$ is given by

$$V[\hat{N}_Y] = \frac{1 - q - (1-q)^{N+1}}{q^2}, \tag{16}$$

where we continue to suppress i in our notation. Note that $V[\hat{N}_Y] < (1-q)/q^2$, i.e., the variance is bounded from above in N. The reason for this advantageous behavior is that, as N grows, the value of the estimator $\hat{N}_Y$ becomes increasingly dominated by the value Y, a quantity that embodies exact knowledge.

THEOREM 4. *$V[\hat{N}_Y] \le V[\hat{N}_X]$, with equality holding only if $N = 0$ or 1.*

PROOF. Starting with the well-known Bernoulli inequality

$$(1-q)^N \ge 1 - Nq, \tag{17}$$

we find that

$$N \ge \frac{1 - (1-q)^N}{q}.$$

The desired result follows after multiplying both sides of the above inequality by $(1-q)/q$, and observing that equality holds in (17) only when N equals 0 or 1. □

For fixed q, we have $V[\hat{N}_Y]/V[\hat{N}_X] \approx 1/(qN)$ as N becomes large, and the variance reduction can be substantial.

Following standard statistical practice, we can estimate $V[\hat{N}_Y]$ using the (biased) estimator

$$\hat{V}[\hat{N}_Y] = \frac{1 - q - (1-q)^{\hat{N}_Y + 1}}{q^2},$$

The bias of the estimator converges to 0 as $q \to 1$.

4.2 Estimating Sums, Averages, and Ratios

The foregoing results for frequencies lead immediately to unbiased estimators for sums and averages, as well as estimator for ratios. In particular, suppose we are given a function $g : T \mapsto \Re$ and we wish to estimate the sum of $g(t)$ over all items t in the dataset. That is, we wish to estimate $\alpha(g) = \sum_{t \in T} g(t)N(t)$. (Again, we have suppressed the subscript i in our notation.) The standard HT estimator of $\alpha(g)$ is

$$\hat{\alpha}_X(g) = \sum_{t \in D(S)} g(t)\hat{N}_X(t) = \sum_{t \in T} g(t)\hat{N}_X(t),$$

where $\hat{N}_X(t)$ corresponds to the estimator $\hat{N}_X$ described in Section 4.1, evaluated with respect to item t, and $D(S)$ denotes the set of distinct items in the sample. The linearity of the expectation operator immediately implies that $E[\hat{\alpha}_X(g)] = \alpha(g)$, so that $\hat{\alpha}_X$ is unbiased. Because items are sampled independently, the estimators $\{\hat{N}_X(t) : t \in T\}$ are mutually independent, and

$$V\big[\hat{\alpha}_X(g)\big] = \sum_{t \in T} g^2(t)\frac{(1-q)N(t)}{q}.$$

Similarly, an improved estimator is given by

$$\hat{\alpha}_Y(g) = \sum_{t \in D(S)} g(t)\hat{N}_Y(t).$$

It follows from Theorem 3 that $\hat{\alpha}_Y(g)$ is unbiased, and, by (16),

$$V\big[\hat{\alpha}_Y(g)\big] = \sum_{t \in T} g^2(t)\frac{1-q-(1-q)^{N(t)+1}}{q^2}.$$

Theorem 4 implies that $V\big[\hat{\alpha}_Y(g)\big] \le V\big[\hat{\alpha}_X(g)\big]$. We can obtain a natural (biased) estimator of $V\big[\hat{\alpha}_Y(g)\big]$ as

$$\hat{V}\big[\hat{\alpha}_Y(g)\big] = \sum_{t \in D(S)} g^2(t)\frac{1-q-(1-q)^{\hat{N}_Y(t)+1}}{q^2\big(1-(1-q)^{\hat{N}_Y(t)}\big)}.$$

This estimator is "almost" the HT estimator of $V\big[\hat{\alpha}_Y(g)\big]$, except that $N(t)$ is replaced by its estimate $\hat{N}_Y(t)$ in each term of the sum.

The foregoing results extend in a straightforward way to averages of the form $\mu = (1/|R|)\sum_{t \in T} g(t)N(t)$, where $|R| = \sum_{t \in T} N(t)$. Since $|R|$ is usually known in applications, it can be treated as a deterministic constant, adding a multiplicative factor of $1/|R|$ to the estimators and a factor of $1/|R|^2$ to the variances and variance estimators. A less trivial scenario arises when estimating a ratio of the form

$$\rho = \frac{\sum_{t \in T} g(t)N(t)}{\sum_{t \in T} h(t)N(t)} = \frac{\alpha(g)}{\alpha(h)},$$

where g and h are arbitrary real-valued functions on T. As special case of such an estimator, take h to be a 0/1 function that corresponds to a predicate defined over T, and take $g(t) = g^*(t)h(t)$, where g^* is an arbitrary real-valued function on T. Then ρ corresponds to the average value of g^* over those elements of T that satisfy the predicate corresponding to h. The ratio estimation problem has been extensively studied [13], and an exhaustive discussion is beyond the scope of the current paper; we content ourselves here with briefly presenting some of the most pertinent results. The usual ratio estimator

$$\hat{\rho}_\Theta = \frac{\hat{\alpha}_\Theta(g)}{\hat{\alpha}_\Theta(h)} = \frac{\sum_{t \in D(S)} g(t)\hat{N}_\Theta(t)}{\sum_{t \in D(S)} h(t)\hat{N}_\Theta(t)},$$

where Θ equals X or Y, is biased, but the bias converges to 0 as q increases. A number of schemes have been proposed to reduce the bias when q is very small; see [13]. When q is not too small, a Taylor-series argument yields an approximate expression for $V[\hat{\rho}_\Theta]$:

$$V[\hat{\rho}_\Theta] \approx \frac{1}{\alpha^2(h)}\Big(V\big[\hat{\alpha}_\Theta(g)\big] + \rho^2 V\big[\hat{\alpha}_\Theta(h)\big] - 2\rho C\big[\hat{\alpha}_\Theta(g), \hat{\alpha}_\Theta(h)\big]\Big),$$

where $C[W,Z]$ denotes the covariance of random variables W and Z. Note that, using the independence of the sampling for different distinct items, we have

$$C\big[\hat{\alpha}_\Theta(g), \hat{\alpha}_\Theta(h)\big] = \sum_{t \in T} g(t)h(t)V\big[\hat{N}_\Theta(t)\big]$$

which can be estimated by

$$\hat{C}\big[\hat{\alpha}_\Theta(g), \hat{\alpha}_\Theta(h)\big] = \sum_{t \in T} g(t)h(t)\hat{V}\big[\hat{N}_\Theta(t)\big].$$

The usual method for estimating the variance $V[\hat{\rho}_\Theta]$ simply replaces ρ by $\hat{\rho}_\Theta$, $\alpha(h)$ by $\hat{\alpha}_\Theta(h)$, C by $\hat{C}$, and each V by $\hat{V}$ in the formula for $V[\hat{\rho}_\Theta]$. Of course, we can always obtain standard errors or estimators of standard errors by taking the square root of the corresponding variances or estimated variances.

4.3 Estimating Distinct-Item Counts

In this section we show that, perhaps surprisingly, an augmented Bernoulli sample can also be used to estimate the number of distinct items in R. Database applications of distinct-item estimation include data integration, query optimization, network monitoring, and OLAP. Although an augmented Bernoulli sample will probably not perform as well as synopses that are designed specifically for this task—see [9] for an overview of specialized methods—the techniques in this section can be useful when special-purpose synopses are not available.

Given an augmented Bernoulli sample S, define an (ordinary) random subset $S' \subseteq T$ by examining each $t \in D(S)$ and including t in S' with probability $p(t)$, where $p(t) = 1$ if $Y(t) = 1$ and $p(t) = q$ if $Y(t) > 1$. Denote by $D(R)$ the set of distinct items in R.

THEOREM 5. *The random subset S' is a BERN(q) sample of $D(R)$.*

PROOF. Fix an item t and observe that $P(t \in S') = 0$ if t is not in S, and hence if $Y(t) = 0$. Then, writing Y for $Y(t)$ and using (6),

$$\begin{aligned} P(t \in S') &= P(Y=1) + qP(Y>1) \\ &= P(Y=1) + q\big(P(Y>0) - P(Y=1)\big) \\ &= (1-q)P(Y=1) + qP(Y>0) \\ &= (1-q)q(1-q)^{N-1} + q\big(1-(1-q)^N\big) = q \end{aligned}$$

Since items are included or excluded from S' independently of each other, the desired result holds. □

We can therefore obtain an unbiased estimate of $D = |D(R)|$, the number of distinct items in R, by using a standard HT estimator, namely, $\hat{D}_{\mathrm{HT}} = |S'|/q$. The variance of this estimator is $(1-q)D/q$ and can be estimated by $(1-q)\hat{D}_{\mathrm{HT}}/q$.

We propose a different unbiased estimator here, which estimates D directly from S and yields lower variance. The idea is to apply the "conditional Monte Carlo principle," which asserts that, if W is an unbiased estimator of some unknown parameter θ and Z is a random variable that represents "additional information," then the random variable $W' = E[W \mid Z]$ is a better estimator than W in that, by the law of total expectation, $E[W'] = E\big[E[W \mid Z]\big] = E[W] = \theta$,

and, moreover, $V[W'] \le V[W]$. The variance reduction is a consequence of the well known and easily derived variance decomposition

$$V[W] = E\big[V[W \mid Z]\big] + V[E[W \mid Z]\big],$$

which holds for any two random variables W and Z. To apply this principle, we take $W = \hat{D}_{\mathrm{HT}}$ and $Z = S$, so that our proposed estimator is $\hat{D}_Y = E[\hat{D}_{\mathrm{HT}} \mid S]$. We can express $\hat{D}_Y$ in a more tractable form, as follows. Denote by $I(A)$ the indicator variable that equals 1 if event A occurs and equals 0 otherwise. Then

$$\begin{aligned}\hat{D}_Y &= E[\hat{D}_{\mathrm{HT}} \mid S] = E\Big[\frac{1}{q}\sum_{t\in D(R)} I(t \in S') \;\Big|\; S\Big] \\ &= \frac{1}{q}\sum_{t\in D(R)} P(t \in S' \mid S) = \frac{1}{q}\sum_{t\in D(R)} I\big(t \in D(S)\big)P\big(t \in S' \mid Y(t)\big) \\ &= \frac{1}{q}\sum_{t\in D(R)} I\big(t \in D(S)\big)p(t) = \sum_{t\in D(S)} p(t)/q.\end{aligned}$$

Intuitively, each distinct item in S is included in S' with probability $p(t)$ and the estimator $\hat{D}_{\mathrm{HT}}$ estimates D by counting all included items followed by a division by q. On average, every item contributes a quantity of $p(t)/q$ to the distinct-item count, and $\hat{D}_Y$ simply adds up the expected contributions. As shown above, $\hat{D}_Y$ is unbiased for D. Moreover, the variance of this estimator can be explicitly computed as

$$\begin{aligned}V[\hat{D}_Y] &= \sum_{t\in D(R)} (1-q)^{N(t)}/q \le \sum_{t\in D(R)} (1-q)/q \\ &= (1-q)D/q = V[\hat{D}_{\mathrm{HT}}],\end{aligned}$$

where equality holds if and only if $D = |R|$. In the usual way, the variance of $\hat{D}_Y$ can be estimated by the "almost" HT estimator

$$\hat{V}[\hat{D}_Y] = \sum_{t\in D(S)} \frac{(1-q)^{\hat{N}_Y(t)}}{q\big(1-(1-q)^{\hat{N}_Y(t)}\big)},$$

where $\hat{N}_Y(t)$ is the estimator of $N(t)$ given in Section 4.1. Note that the memory requirement for our technique is unbounded, unlike the estimator in [8] and other specialized distinct-item estimation methods as in [9].

5. SUBSAMPLING

The subsampling problem can be described as follows: given an augmented Bernoulli sample S of a dataset R drawn with sampling rate q, derive an augmented Bernoulli sample S' from R with sampling rate $q' \le q$ without accessing R. Subsampling has applications in practice whenever the sampling process is run on a system with bounded processing or space capabilities. In more detail, whenever some criterion is satisfied, one may reduce the sampling rate in order to effectively reduce the size of the sample. For example, several of the techniques in [2] use Bernoulli subsampling to enforce an upper bound on the sample footprint.

As indicated in Section 2.1, the subsampling problem is trivial if we are only trying to produce an ordinary Bernoulli subsample that does not need to be incrementally maintained. The challenge in the general setting is to assign an appropriate value to the tracking counter of the subsample, so that incremental maintenance can be continued.

We now present a subsampling algorithm. Let S_i be the sample after processing transactions $\gamma_1, \ldots, \gamma_i$ with sampling rate q. Again, we suppress the subscript i and fix an item t, so that S_i is given by $S = (X,Y)$. Given S, we want to generate $S' = (X',Y')$ having the correct distribution.

The algorithm is as follows: Set $q^* = q'/q$. Let Φ be a Bernoulli(q^*) random variable, i.e., a 0/1 random variable with $P(\Phi = 1) = q^*$ and $P(\Phi = 0) = 1 - q^*$. Let Ψ be a random variable such that $P(\Psi = k') = B(k'; X-1, q^*)$ for $0 \le k' < X$. Observe that $\Psi \equiv 0$ when $X = 1$. The random variable Φ has the interpretation that $\Phi = 1$ if and only if the first of the X items that were inserted into S is retained in S'; the random variable Ψ is the number of the remaining $X-1$ items that are retained. The algorithm sets

$$X' = \begin{cases} 0 & \text{if } X = 0 \\ \Phi + \Psi & \text{otherwise.} \end{cases} \tag{18}$$

To compute Y', let Υ be another random variable with

$$P(\Upsilon = m') = \frac{X'}{m'} \prod_{i=m'+1}^{Y-1} \left(1 - \frac{X'}{i}\right)$$

for $X' \le m' < Y$. (By convention, we take an empty product as equal to 1.) The algorithm sets

$$Y' = \begin{cases} 0 & \text{if } X' = 0 \\ Y & \text{if } X' > 0 \text{ and } \Phi = 1 \\ \Upsilon & \text{otherwise.} \end{cases} \tag{19}$$

We now establish the correctness of the subsampling algorithm.

THEOREM 6. *The above subsampling algorithm produces an augmented Bernoulli sample with sampling rate q'.*

PROOF. We must show that the random pair (X',Y') has the proper distribution, given (X,Y), after determining what this proper distribution is. As with Theorem 2, we give a proof that is somewhat informal, but provides insight into the workings of the algorithm. Denote by $\gamma = (\gamma_1, \ldots, \gamma_N)$ a sequence of N insertions of item t. Since, by Theorem 2, the distribution of X and Y depend only on N, we can assume without loss of generality that S has been generated from γ. (I.e., S is based on N non-annihilated insertions.)

First consider the distribution of X'. Clearly, X' must equal 0 if X equals 0. Otherwise, as outlined in Section 2.1, the correct way to obtain X' from X is to take a BERN(q^*) subsample of the X items in S. Therefore, the random variable X' must have a Binomial(X,q^*) distribution, i.e., $P(X' = k') = B(k'; X, q^*)$. Recall that, in general, a Binomial(m,q) random variable can be represented as the sum of m independent and identically distributed (i.i.d.) Bernoulli(q) random variables. Thus Ψ can be viewed as a sum of $X-1$ i.i.d. Bernoulli(q) random variables, so that $\Phi + \Psi$ is distributed as a sum of X such variables, and hence has a Binomial(X,q^*) distribution. Therefore X', as defined by (18), indeed has the proper distribution.

To complete the proof, it suffices to show that the conditional distribution of Y' is the proper one, given that we have taken a BERN(q^*) subsample that resulted in X' items being retained. To determine the proper distribution, first observe that, trivially, Y' must equal 0 if X' equals 0. To analyze the case where $X' > 0$, let $j^* = N - Y + 1$ be the index of the transaction corresponding to the first insertion of an item into S; thus the remaining $X-1$ items in S were inserted during transactions $\gamma_{j^*+1}, \gamma_{j^*+2}, \ldots, \gamma_N$. Clearly, $Y' = Y$ if and only if the item inserted into S at transaction γ_{j^*} is retained in the subsample S'; this event occurs with probability q^*. With probability $1 - q^*$, we have $Y' \ne Y$. In this case, we can compute the proper distribution of Y' as follows. The transactions corresponding to the X' items in the final subsample represent a subset (of size X') of the $Y-1$ transactions $\gamma_{j^*+1}, \gamma_{j^*+2}, \ldots, \gamma_N$. By symmetry, all possible transaction-subsets of size X' are equally likely. Thus, if we have an urn containing $Y-1$ balls, of which

X' are black and $Y-1-X'$ are white, and if Z is the random variable that represents the number of sequential draws (without replacement) required to produce the first black ball, then the first of the X' transactions has an index distributed as j^*+Z, so that Y' must be distributed as $N-(j^*+Z)+1=Y-Z$. For Z to equal $l>1$, the first ball must be white, which happens with probability $1-\big(X'/(Y-1)\big)$, then the next ball must be white, which happens with probability $1-\big(X'/(Y-2)\big)$ (since there is one less white ball in the urn after the first draw), and so on, up through the $(l-1)$st ball; the lth ball must be black, which happens with probability $X'/(Y-l)$. Similarly, $Z=1$ with probability $X'/(Y-1)$. Thus

$$P(Z=l) = \frac{X'}{Y-l}\prod_{i=1}^{l-1}\left(1-\frac{X'}{Y-i}\right) = \frac{X'}{Y-l}\prod_{i=Y-l+1}^{Y-1}\left(1-\frac{X'}{i}\right)$$

for $l \geq 1$, where the second equality results after a change of index from i to $Y-i$ in the product. Since $Y'=Y-Z=m'$ if and only if $Z=Y-m'$, then we must have

$$P(Y'=m' \mid Y' \neq Y) = P(Z=Y-m') = \frac{X'}{m'}\prod_{i=m'+1}^{Y-1}\left(1-\frac{X'}{i}\right).$$

By inspection, the random variable Y' defined via (19) has precisely the correct distribution. As mentioned previously, Φ has the interpretation that $\Phi=1$ if and only if $Y'=Y$, i.e., if and only if the item inserted into S at transaction γ_{j^*} is retained in S'. □

6. MERGING

We now define the merging problem. Given partitions R_1 and R_2 of R with $R_1 \cup R_2 = R$, along with two corresponding augmented Bernoulli samples S_1 and S_2—mutually independent and each with sampling rate q—derive an augmented Bernoulli sample S from R with sampling rate q by accessing S_1 and S_2 only.[3] Merging is used in practice when R is distributed across several nodes; see [2] for an example. One may then compute a sample of each partition locally and derive a sample of the complete dataset by merging these local samples. This approach is often superior, in terms of parallelism and communication cost, to first reconstructing R and sampling afterwards. The merging operation can be used more generally to obtain a sample of any specified union of the partitions [2].

If S is not subject to further maintenance, simply set $X(t) = X_1(t)+X_2(t)$ for all $t \in S_1 \cup S_2$, thereby producing a non-augmented Bernoulli sample; see Section 2.1. Here $X_1(t)$ and $X_2(t)$ denote the frequency of item t in the respective subsamples, and $X(t)$ denotes the frequency of item t in the merged sample. A harder version of the problem is to derive a merged sample S that includes a tracking counter, so that maintenance of S can be continued. First suppose that we know a priori that $R_1 \cap R_2 = \emptyset$. Then it is easy to show that setting $S = S_1 \cup S_2$ yields the desired augmented Bernoulli sample. Otherwise, the hard merging problem cannot be solved, as shown by the following negative result.

THEOREM 7. *If $R_1 \cap R_2 \neq \emptyset$ and $0 < q < 1$, then there exists no algorithm that can compute an augmented Bernoulli sample S of $R = R_1 \cup R_2$ by accessing S_1 and S_2 only.*

PROOF. The proof is by contradiction, so suppose that there exists a merging algorithm A and let S be the augmented Bernoulli sample produced by A. Fix an item t in the intersection of R_1 and R_2. There is always such an item, since $R_1 \cap R_2 \neq \emptyset$. Denote by N, N_1, and N_2 the frequency of item t in R, R_1, and R_2, respectively. Furthermore, let Y, Y_1, and Y_2 be the value of the tracking counters in S, S_1, and S_2, respectively.

We first show that A cannot ever set $Y > Y_1 + Y_2$. From (6), we must have $P(Y_1 = N_1) = P(Y_2 = N_2) = q$ so that, by the independence of Y_1 and Y_2,

$$P(Y_1 = N_1, Y_2 = N_2) = q^2. \tag{20}$$

Suppose that Algorithm A were to set $Y > Y_1 + Y_2$ with positive probability when it observed, say, input values $Y_1 = k_1$ and $Y_2 = k_2$, that is, $P(Y > k_1 + k_2 \mid Y_1 = k_1, Y_2 = k_2) > 0$. If item t happened to occur exactly k_1 and k_2 times in R_1 and R_2, respectively, so that $N_1 = k_1$ and $N_2 = k_2$, then

$$\begin{aligned} &P(Y > N) \\ &\geq P(Y > k_1 + k_2, Y_1 = k_1, Y_2 = k_2) \\ &= P(Y > k_1 + k_2 \mid Y_1 = k_1, Y_2 = k_2) P(Y_1 = k_1, Y_2 = k_2) \\ &= P(Y > k_1 + k_2 \mid Y_1 = k_1, Y_2 = k_2) q^2 > 0, \end{aligned}$$

where we have used (20). But $Y \in \{0, 1, \ldots, N\}$ by definition, so that Algorithm A cannot ever set $Y > Y_1 + Y_2$.

We now show that A must set $Y > Y_1 + Y_2$ with positive probability, a contradiction. Observe that

$$\begin{aligned} P(Y = N) = P\big(Y = N, (Y_1, Y_2) = (N_1, N_2)\big) \\ + P\big(Y = N, (Y_1, Y_2) \neq (N_1, N_2)\big). \end{aligned} \tag{21}$$

By (6) and (20), we must have

$$\begin{aligned} P(Y = N) = q > q^2 = P\big((Y_1, Y_2) = (N_1, N_2)\big) \\ \geq P\big(Y = N, (Y_1, Y_2) = (N_1, N_2)\big), \end{aligned}$$

which implies that

$$P\big(Y = N, (Y_1, Y_2) \neq (N_1, N_2)\big) > 0 \tag{22}$$

for the equality in (21) to hold. Since $N = N_1 + N_2$ and, by definition, $Y_1 \leq N_1$ and $Y_2 \leq N_2$, the assertion in (22) is equivalent to $P(Y = N, Y_1 + Y_2 < N) > 0$, so that $P(Y > Y_1 + Y_2) > 0$. □

7. IMPLEMENTATION ISSUES

Algorithm 1 gives the pseudocode for our sample maintenance algorithm. In the algorithm, the function RAND generates a pseudorandom number in the interval $[0, 1)$. As observed in [14], one may significantly reduce the number of calls to RAND in line 7 and 12. The idea is to maintain a counter L for the number of items which are skipped before the next sample insertion (line 8 and 13). If $L > 0$ then the item is rejected and L is decremented. Otherwise, $L = 0$, the item is accepted and L is regenerated. By (12), it follows that L is geometrically distributed. Efficient algorithms for generating geometric random variables such as L are given in [4, p. 498].

Algorithm 2 gives the pseudocode for our subsampling algorithm. We have displayed simple versions of the functions COMPUTEΨ and COMPUTEΥ, which generate samples of the random variables Ψ and Υ, respectively. Again, efficient algorithms are given in [4, p. 521] and [4, p. 619], respectively.

8. RELATED WORK

A variety of sample-maintenance methods have been proposed that require access to the underlying dataset; see [12] and the "reservoir sampling with recomputation" method discussed in [7]. As

[3] Here, $\cup$ has multiset semantics, so that it need not be the case that $R_1 \cap R_2 = \emptyset$.

Algorithm 1 Bernoulli sampling with deletions

```
 1: q: the sampling rate
 2: S: the augmented Bernoulli sample
 3:
 4: INSERT(t):
 5: if S contains t then
 6:    (X,Y) ← S[t]
 7:    if RAND() < q then                    // t ∈ S, insertion accepted
 8:       X ← X + 1
 9:    end if
10:    Y ← Y + 1
11:    S[t] ← (X,Y)
12: else if RAND() < q then                  // t ∉ S, insertion accepted
13:    S[t] ← (1,1)                          // add t to the sample
14: end if
15:
16: DELETE(t):
17: if S contains t then
18:    (X,Y) ← S[t]
19:    if Y = 1 then                         // last seen occurrence of t
20:       remove t from S
21:    else                                  // more than one occurrence
22:       if RAND() < (X−1)/(Y−1) then       // deletion accepted
23:          X ← X − 1
24:       end if
25:       Y ← Y − 1
26:       S[t] = (X,Y)
27:    end if
28: end if
```

Algorithm 2 Subsampling

```
 1: q: current sampling rate
 2: q′: desired sampling rate, q′ ≤ q
 3: S: the augmented Bernoulli sample
 4:
 5: for all t ∈ S do
 6:    (X,Y) ← S[t]
 7:    X ← COMPUTEΨ(X, q*)
 8:    if RAND() < q* then                   // first item included
 9:       X ← X + 1
10:       S[t] = (X,Y)
11:    else if X = 0 then                    // all items excluded
12:       remove t from S
13:    else                                  // first item excluded
14:       Y ← COMPUTEϒ(X,Y)
15:       S[t] = (X,Y)
16:    end if
17: end for
18:
19: COMPUTEΨ(X, q*)                          // simple version
20: Ψ ← 0
21: for 1 ≤ i ≤ X − 1 do
22:    if RAND() < q* then
23:       Ψ ← Ψ + 1
24:    end if
25: end for
26: return Ψ
27:
28: COMPUTEϒ(X,Y)                           // simple version
29: ϒ ← Y − 1
30: while RAND() < X/ϒ do
31:    ϒ ← ϒ − 1
32: end while
33: return ϒ
```

discussed previously, these methods tend to be impractically expensive.

The oldest of those maintenance methods that do not require dataset access is the classical reservoir sampling algorithm originally proposed in [5] and streamlined in [14]. These algorithms maintain bounded uniform samples and handle only insertion transactions. Babcock, et al. [1] provide a maintenance method that can deal with synchronized insertions and deletions, as encountered in a fixed-size sliding window over a data stream. If the dataset does not contain duplicates (i.e., is a real set), then the "random pairing" algorithm introduced in [7] generalizes both the algorithm in [1] and reservoir sampling to maintain a bounded uniform sample in the presence of arbitrary UDI transactions. The authors also provide a resizing algorithm to grow a uniform sample in a controlled manner, to handle situations in which the underlying dataset is growing; our new maintenance algorithm, which maintains an unbounded Bernoulli sample, handles such dataset-growth issues automatically.

The methods closest to ours are the concise-sample and counting-sample methods introduced in [10]. A concise sample comprises pairs of the form $(t, X_i(t))$, where $X_i(t)$ is defined as in Section 2.2. Concise samples handle only insertion transactions, and are shown in [10] to yield more precise estimates than a sample with the same footprint represented in uncompressed form. A counting sample comprises pairs of the form $(t, Y_i(t))$, where $Y_i(t)$ is defined as in Section 3. The maintenance of a counting sample is similar to the maintenance of our augmented Bernoulli sample. As discussed in [10], a concise sample can be extracted from a counting sample via a coin-flipping step. Our algorithms amortize this subsampling cost over all transactions, thereby facilitating faster on-demand materialization of the sample, and hence faster production of estimates based on the sample. Since we maintain the value of X_i directly instead of randomly generating it every time the sample is accessed, we expect that estimates derived from our augmented Bernoulli sample are more stable statistically, especially when they are computed frequently. Counting samples were originally introduced for estimating the frequencies of "hot" (i.e., frequent) items. The authors provided the estimator $Y_i(t) - 1 + 0.418/q$ for estimating the frequency of hot item t. As can be seen from Theorem 3, this estimator is biased, has the same variance as $\hat{N}_Y(t)$, and therefore has a higher mean squared error (MSE). We provide unbiased, low-MSE estimators for the frequencies of all items, not just hot items. Another key difference between our current results and those in [10] is that the concise-sample and counting-sample methods use subsampling to ensure a bounded sample footprint; because such subsampling has been shown to compromise the uniformity of the samples [2], we avoid this technique. The current paper shows how to combine counting samples and concise samples to seamlessly maintain a true Bernoulli sample under arbitrary UDI transactions, and how to estimate not just frequencies, but population totals, averages and distinct-item counts in an unbiased manner, with low variance. Note that the new estimators described in Section 4 can also be used with counting samples.

An orthogonal approach is taken by the distinct-item (DI) sampling schemes given in [3, 6, 8]. These schemes are primarily designed for estimating quantities such as the number of distinct values in a dataset [8] or the fraction of distinct items that have a specified frequency [3]. These algorithms maintain, in effect, a simple random sample of $\big(t, N(t)\big)$ pairs; the frequency $N(t)$ is either rep-

resented exactly, or via a high-accuracy approximation. The probability that t appears in the DI sample is independent of its frequency $N(t)$; low-frequency items are over-represented in the DI sample relative to Bernoulli samples, whereas high-frequency items are under-represented. Therefore, such samples are not well suited to estimating frequencies of individual items, especially highly-frequent items. DI samples can potentially be used to estimate sums of the form $\alpha(g)$, as defined in Section 4.2. Note, however, that such estimates involve a scale-up factor of $D(R)/D(S)$, where $D(R)$ and $D(S)$ are the numbers of distinct items in R and S. Thus, unlike for our estimators, $D(R)$ needs to be known or estimated. Assuming that this issue can be dealt with, a DI estimator can be advantageous, for example, when $g(t)$ is small for most $t \in T$ but is very large for a particular item t' having very low frequency, since this rare anomalous item is more likely to be included into a DI sample than into a Bernoulli sample. On the other hand, if the values $\{g(t) : t \in T\}$ are relatively homogeneous but the item frequencies are highly skewed, then the highly frequent values that dominate the value of $\alpha(g)$ are more likely to be excluded from a DI sample than from a Bernoulli sample, leading to estimation errors. In any case, the potential advantages of DI samples as in [3, 6] are often offset by the very high space overhead required for sample maintenance relative to Bernoulli or augmented Bernoulli samples [7]. I.e., for a given space allocation, we can maintain a significantly larger Bernoulli sample.

9. SUMMARY

We have provided a scheme for maintaining a Bernoulli sample of an evolving multiset. Our maintenance algorithm can handle arbitrary UDI transactions, and avoids ever accessing the underlying dataset. We have shown that the tracking counters used for maintenance can also be exploited to estimate frequencies, population sums, population averages, and distinct-item counts in an unbiased manner, with variance lower (often much lower) than the standard estimates based on a Bernoulli sample. We have also indicated how to estimate the variance (and hence the standard error) for these estimates, and have briefly described how to apply results from ratio-estimation theory to our new estimators. We have also described how to obtain an augmented sample from another such sample using subsampling, and have identified the (rather limited) conditions under which augmented samples can be merged to obtain an augmented sample of the union of the underlying datasets. Fortunately, in practice it often suffices to maintain individual augmented samples, occasionally providing (ordinary) merged samples on demand.

10. REFERENCES

[1] B. Babcock, M. Datar, and R. Motwani. Sampling from a moving window over streaming data. In *Proc. SODA*, pages 633–634, 2002.

[2] P. G. Brown and P. J. Haas. Techniques for warehousing of sample data. In *Proc. ICDE*, 2006.

[3] G. Cormode, S. Muthukrishnan, and I. Rozenbaum. Summarizing and Mining Inverse Distributions on Data Streams via Dynamic Inverse Sampling. In *Proc. VLDB*, pages 25–36, 2005.

[4] L. Devroye. *Non-Uniform Random Variate Generation*. Springer, New York, 1986.

[5] C. Fan, M. Muller, and I. Rezucha. Development of sampling plans by using sequential (item by item) techniques and digital computers. *J. Amer. Statist. Assoc.*, 57:387–402, 1962.

[6] G. Frahling, P. Indyk, and C. Sohler. Sampling in dynamic data streams and applications. In *Proc. 21st Annual Symp. Comput. Geom.*, pages 142–149, 2005.

[7] R. Gemulla, W. Lehner, and P. J. Haas. A dip in the reservoir: maintaining sample synopses of evolving datasets. In *Proc. VLDB*, pages 595–606, 2006.

[8] P. B. Gibbons. Distinct sampling for highly-accurate answers to distinct values queries and event reports. In *Proc. VLDB*, pages 541–550, 2001.

[9] P. B. Gibbons. Distinct-values estimation over data streams. In *Data Stream Management: Processing High Speed Data Streams*. Springer, 2007.

[10] P. B. Gibbons and Y. Matias. New sampling-based summary statistics for improving approximate query answers. In *Proc. ACM SIGMOD*, pages 331–342, 1998.

[11] C. Jermaine, A. Pol, and S. Arumugam. Online maintenance of very large random samples. In *Proc. ACM SIGMOD*, pages 299–310, 2004.

[12] F. Olken and D. Rotem. Simple random sampling from relational databases. In *Proc. VLDB*, pages 160–169, 1986.

[13] C-E. Särndal, B. Swensson, and J. Wretman. *Model Assisted Survey Sampling*. Springer, New York, 1992.

[14] J.S. Vitter. Random Sampling with a Reservoir. *ACM Trans. Math. Software*, 11(1):37–57, 1985.