Estimating Statistical Aggregates on Probabilistic Data Streams

T. S. JAYRAM
IBM Almaden Research
and
ANDREW MCGREGOR
University of California, San Diego
and
S. MUTHUKRISHNAN
Google, Inc.
and
ERIK VEE
Yahoo! Research

The probabilistic stream model was introduced by Jayram, Kale, and Vee [2007]. It is a generalization of the data stream model that is suited to handling "probabilistic" data, where each item of the stream represents a probability distribution over a set of possible events. Therefore, a probabilistic stream determines a distribution over a potentially exponential number of classical "deterministic" streams where each item is deterministically one of the domain values.

We present algorithms for computing commonly used aggregates on a probabilistic stream. We present the first one pass streaming algorithms for estimating the expected mean of a probabilistic stream. Next, we consider the problem of estimating frequency moments for probabilistic data. We propose a general approach to obtain unbiased estimators working over probabilistic data by utilizing unbiased estimators designed for standard streams. Applying this approach, we extend a classical data stream algorithm to obtain a one-pass algorithm for estimating F_2, the second frequency moment. We present the first known streaming algorithms for estimating F_0, the number of distinct items on probabilistic streams. Our work also gives an efficient one-pass algorithm for estimating the median and a two-pass algorithm for estimating the range.

Categories and Subject Descriptors: F.2.0 [**Analysis of Algorithms and Problem Complexity**]: General

General Terms: Algorithms, Theory

Additional Key Words and Phrases: probabilistic streams, OLAP, mean, median, frequency moments

Authors' addresses: T. S. Jayram, IBM Almaden Research, Almaden, CA; A. McGregor, Information Theory and Applications Center, University of California, La Jolla, CA 92093; S. Muthukrishnan, Google Research, New York, NY; Erik Vee, Yahoo! Research, Sunnyvale, CA.
Part of this work was done when the fourth author was at IBM Almaden Research Center

1. INTRODUCTION

In many applications it is becoming increasingly necessary to consider uncertain data, e.g., information that is incomplete, unreliable, or noisy. In this paper, we focus on efficiently calculating aggregation measures over uncertain data, with particular motivation from the OLAP model. OLAP is a multidimensional model of data in which the dimensions are structured hierarchically, and *facts* map to points in the corresponding multidimensional space. For example, we may have a database on the location and costs for different car repairs. The location field is a hierarchy: Santa Clara is a region (city) within California, which is itself a region (state) within the United States. Likewise, the car field is a hierarchy: Civic LX is a Civic, which is a Honda. A fact then specifies a particular point, such as a repair on a Honda Civic LX done in Santa Clara for $350. Dealing with uncertain information in OLAP is widely recognized to be an important problem and has received increasing attention recently. A particular kind of uncertainty, called *imprecision*, arises when facts do not map to points but to regions consistent with the domain hierarchies associated with dimension attributes. For example, a fact might now denote a repair done on a Honda in California for $500. Answering the simple query, "What is the average cost of a car repair in Santa Clara?" then becomes much less clear-cut. Should we include the general fact in our calculations, and to what degree?

Motivated by several criteria, [Burdick et al. 2005a] proposed the following solution. Using a mathematically principled approach, each imprecise "region" fact r in the database is replaced with a set of precise "point" facts representing the possible completions of r, together with their probabilities. For example, a fact about California might be replaced by 3 facts: one about Los Angeles, one about San Diego, and one about San Francisco, with associated probabilities $0.3, 0.2, 0.5$, respectively. This defines a sample space of possible worlds; the probability associated with each world can be computed easily. The answer to an aggregation query Q is then defined to be the expected value of Q over the possible worlds. A similar data model was considered earlier by [Fuhr and Roelleke 1997] in the context of non-aggregation queries, such as conjunctive queries, wherein an algebra of atomic events is used to define the semantics of queries. However, the data (expression) complexity of evaluating conjunctive queries in this framework has been shown to be #P-complete [Dalvi and Suciu 2004].

To answer queries *efficiently*, we cannot explicitly enumerate all the possible worlds, since in the worst case, this could be exponentially large in the size of the database. A more significant challenge while dealing with OLAP queries arises when aggregation queries are qualified by specifying the dimension values at some level of granularity, e.g., average of all repair costs in San Francisco. Here, even the set of facts that match the query specification is a varying quantity over the possible worlds. Furthermore, because of the massive amounts of data being analyzed, it is impractical to process the data using many random accesses, necessitating algorithms that work over *probabilistic streams.*

1.1 Probabilistic Streams

In order to deal with massive data sets that arrive online and have to be monitored, managed and mined in real time, the *data stream* model has become popular. In

stream A, new item a_t that arrives at time t is from some universe $[n] = \{1, \ldots, n\}$. Applications need to monitor various aggregate queries such as quantiles, number of distinct items, and heavy-hitters, in small space, typically $O(\text{polylog}\, n)$. Data stream management systems are becoming mature at managing such streams and monitoring these aggregates on high speed streams such as in IP traffic analysis [Cranor et al. 2003], financial and scientific data streams [Balazinska et al. 2004], and others. See [Babcock et al. 2002; Muthukrishnan 2006] for surveys of systems and algorithms for data stream management.

A generalization of the model, the *probabilistic stream model*, was recently introduced [Jayram et al. 2007]. Here, each item a_t is not an element of the universe, but represents a distribution over elements of the universe together with a probability that the element is not actually present in the stream. The probabilistic stream model is applicable in a variety of settings in addition to the OLAP model. For example, a probabilistic stream may be derived from a stream of queries to a search-engine. Here a query "Stunning Jaguar" would be associated with different topics of interest, such as "Car", "Animal", or "Operating System", with different probabilities. There are also instances when the input stream is probabilistic by its nature: for example, when we measure physical quantities such as light luminosity, temperature and others, these measurements are taken over many tiny time instants in the analog world and are inherently noisy. And in some applications, these measurements are even noisier due to frequent sensor failures and calibration errors.

Previous work on probabilistic streams has included data stream algorithms for certain aggregate queries [Burdick et al. 2005a; 2006; Jayram et al. 2007]. In this paper, we improve those results and study additional aggregate functions which are fundamental in any stream monitoring scenario. We note that some related results for estimating the expectation and variance of the number of distinct values and join/self-join sizes of a probabilistic stream have recently been proven independently of our work [Cormode and Garofalakis 2007]. In what follows, we will first describe the model precisely and then state our results.

1.2 The Model and Definitions

In this section, we formally define the probabilistic stream model and the aggregate statistics that we will study.

Definition 1 Probabilistic Stream. *A probabilistic stream is a data stream $A = \langle \vartheta_1, \vartheta_2, \ldots, \vartheta_m \rangle$ in which each data item ϑ_i encodes a random variable that takes a value in $[n] \cup \{\perp\}$. In particular, each ϑ_i consists of a set of at most l tuples of the form $(j, p_i(j))$ for some $j \in [n]$ and $p_i(j) \in [0, 1]$. These tuples define the random variable X_i where $X_i = j$ with probability $p_i(j)$ and $X_i = \perp$ otherwise. We define $p_i(\perp) = \Pr[X_i = \perp]$ and $p_i(j) = 0$ if not otherwise determined.*

Here, we use $\perp$ to denote the null event. For example, when making a query about repair costs for cars in Santa Clara, a given fact in our database may indicate that with probability 0.60, a repair cost $500, and with probability 0.40, no repair happened (denoted by $\perp$); often, this absence of a repair entry occurs because the repair happened in a location not specified in the query, say in Los Angeles.

A probabilistic stream naturally gives rise to a distribution over "deterministic" streams. Specifically we consider the ith element of the stream to be determined

according to the random variable X_i and each element of the stream to be determined independently. Hence,

$$\Pr[\langle x_1, x_2, \ldots, x_m\rangle] = \prod_{i\in[m]} \Pr[X_i = x_i] = \prod_{i\in[m]} p_i(x_i) \ .$$

Throughout this paper we assume each probability specified in the probability stream is either 0 or $\Omega(1/\operatorname{poly}(n))$.

We will be interested in the expected value of various quantities of the deterministic stream induced by a probabilistic stream. Computing the expected value is intuitively justified, since it is a natural way to summarize the answers of computing Q over an exponential number of possible streams. A formal justification using desiderata for handling imprecision in OLAP is given in [Burdick et al. 2005a].

Definition 2 Aggregation Queries. *We consider the following aggregates:*

(1) $\mathsf{SUM} = \mathbb{E}\left[\sum_{i\in[m]:X_i\neq\perp} X_i\right]$

(2) $\mathsf{COUNT} = \mathbb{E}\left[|\{i \in [m] : X_i \neq \perp\}|\right]$

(3) $\mathsf{MEAN} = \mathbb{E}\left[\sum_{i\in[m]:X_i\neq\perp} X_i / |\{i \in [m] : X_i \neq \perp\}| \,\middle|\, |\{i \in [m] : X_i \neq \perp\}| \neq 0\right]$

(4) $\mathsf{DISTINCT} = \mathbb{E}\left[|\{j \in [n] : \exists i \in [m],\ X_i = j\}|\right]$

(5) $\mathsf{REPEAT\text{-}RATE} = \mathbb{E}\left[\sum_{j\in[n]} |\{i \in [m] : X_i = j\}|^2\right]$

(6) $\mathsf{MEDIAN} = x$ *such that*

$$\lceil \mathsf{COUNT} \rceil / 2 \geq \max\{\mathbb{E}\left[|\{i \in [m] : X_i < x\}|\right], \mathbb{E}\left[|\{i \in [m] : X_i > x\}|\right]\} \ .$$

(7) $\mathsf{RANGE} = \mathbb{E}\left[\max_i(X_i) - \min_i(X_i) + 1\right]$.[1]

These are fundamental aggregates for stream monitoring, and well-studied in the classical stream model. For example, REPEAT-RATE in the classical model is the F_2 estimation problem studied in the seminal [Alon et al. 1999]. DISTINCT (also known as F_0) and MEDIAN have a long history in the classical data stream model [Munro and Paterson 1980; Flajolet and Martin 1985]. In the probabilistic stream model, we do not know of any prior algorithm with guaranteed bounds for these aggregates except the MEAN (SUM and COUNT are trivial) which, as we discuss later, we will improve.

As in classical data stream algorithms, our algorithms need to use polylog(n) space and time per item, on a single pass over the data. It is realistic to assume that $l = O(\operatorname{polylog} n)$ so that processing each item in the stream is still within this space and time bounds. In most motivating instances, l is likely to be quite small, say $O(1)$. In fact, it was shown in [Jayram et al. 2007] that $l = 1$ is sufficient for OLAP queries. Also, since the deterministic stream is an instance of the probabilistic stream (with each item in the probabilistic stream being one deterministic item with probability 1), known lower bounds for the deterministic stream model carry over to the probabilistic stream model. As a result, for most estimation problems, there does not exist streaming algorithms that precisely calculate the aggregate.

[1] While the range of continuous data is usually defined as the maximum value minus the minimum value, since our data takes values from $[n]$, we include a "+1" term such that $1 \leq \mathsf{RANGE} \leq n$.

	Space	Randomized/Deterministic
MEAN	$O(\log \varepsilon^{-1})$	Deterministic
DISTINCT	$O((\varepsilon^{-2} + \log n) \log \delta^{-1})$	Randomized
REPEAT-RATE	$O(\varepsilon^{-2}(\log n + \log m) \log \delta^{-1})$	Randomized
MEDIAN	$O(\varepsilon^{-2} \log m)$	Deterministic
RANGE	$O(\varepsilon^{-1} \log m)$	Deterministic

Table I. Streaming Algorithms for Probabilistic Streams. All algorithms are one-pass with the exception of the algorithm for RANGE that requires two passes and requires the assumption that $\mathsf{COUNT} = \Omega(\log(n\varepsilon^{-1}))$.

Instead, we will focus on returning approximations to the quantities above. We say a value $\hat{Q}$ is an (ε, δ)-approximation for a real number Q if $|\hat{Q} - Q| \leq \varepsilon Q$ with probability at least $(1 - \delta)$ over its internal coin tosses. Many of our algorithms will be deterministic and return $(\varepsilon, 0)$-approximations. When approximating MEDIAN it makes more sense to consider a slightly different notion of approximation. We say x is an ε-approximate median if $(1/2 + \varepsilon)\lceil \mathsf{COUNT} \rceil$ is greater than

$$\max\{\mathbb{E}\left[|\{i \in [m] : X_i < x\}|\right], \mathbb{E}\left[|\{i \in [m] : X_i > x\}|\right]\} \ .$$

1.2.1 *Related Models.* The probabilistic stream model complements a stream model that has been recently considered where the stream consists of independent samples drawn from a single unknown distribution [Chang and Kannan 2006; Guha and McGregor 2007]. In this alternative model, the probabilistic stream $A = \langle \vartheta_1, \ldots, \vartheta_1 \rangle$ consists of a repeated element encoding a single probability distribution over $[n]$, but the crux is that the algorithm does not have access to the probabilistic stream. The challenge is to infer properties of the probability distribution ϑ_1 from a randomly chosen deterministic stream. (In this setting it is assumed that l is large.) There is related work on reconstructing strings from random traces [Batu et al. 2004; Kannan and McGregor 2005]. Here, each element of probabilistic stream is of the form $\{(i, 1-p), (\perp, p)\}$ for some $i \in [n]$. As before, the algorithm does not have access to the probabilistic stream but rather tries to infer the probabilistic stream from a limited number of independently generated deterministic streams. We emphasize that the results in [Batu et al. 2004; Chang and Kannan 2006; Guha and McGregor 2007; Kannan and McGregor 2005] do not provide any bounds for estimating aggregates such as the ones we study here when input is drawn from multiple, known probability distributions.

1.3 Our Results

We present the first single-pass algorithms for estimating the aggregate properties MEAN, MEDIAN, REPEAT-RATE, and DISTINCT. The algorithms for MEAN and MEDIAN are deterministic while the other two algorithms are randomized. While it is desirable for all the algorithms to be deterministic, this randomization can be shown to be necessary using the standard results in streaming algorithms, eg., of Alon, Matias, and Szegedy [Alon et al. 1999, Proposition 3.7]. Our results are summarized in the Table I.

At first glance, MEAN seems trivial to estimate and this is the case with deterministic streams. SUM and COUNT can indeed be estimated easily since they are linear in the input and, hence, we can write straightforward formulas to compute them. However MEAN is not simply SUM/COUNT as shown in [Jayram et al. 2007] and needs nontrivial techniques.

We present two single-pass, deterministic, $(\varepsilon, 0)$-approximation algorithms for MEAN. The first using $O(\log \varepsilon^{-1})$ words of space while the second, using an alternative technique, uses $O(\varepsilon^{-1/2})$ space. Furthermore, if COUNT is sufficiently large then we present a simple algorithm that only needs $O(1)$ words of space. We further demonstrate a tradeoff between update time and space, additionally giving a one-pass $(\varepsilon, 0)$-approximation algorithm for MEAN that works in $O(1/\varepsilon)$ words of space, but has update time of just $O(1)$. (The algorithm using $O(\log \varepsilon^{-1})$ words has update time of $O(\log \varepsilon^{-1})$.) The best known previous work [Jayram et al. 2007] presents an $O(\log m)$ pass algorithm using $O(\varepsilon^{-1} \log^2 m)$ space. Thus our algorithms represent a considerable improvement.

Other aggregates such as MEDIAN, REPEAT-RATE, and DISTINCT have previously known algorithms for deterministic streams. A natural approach therefore would be to randomly instantiate multiple deterministic streams, apply standard stream algorithms and then apply a robust estimator. This approach works to an extent but typically gives poor approximations to small quantities. This is the case for DISTINCT and we need to develop an alternative strategy for when DISTINCT is small. For MEDIAN, we show that it is possible to deterministically instantiate a single deterministic stream and run standard stream algorithms.

Our approach for estimating REPEAT-RATE is somewhat different. REPEAT-RATE is the expected value of the second frequency moment (recall that the i-th frequency moment F_i for an input stream of elements from a fixed domain of size n is defined to be the sum of the i-th power of the frequencies of the different elements in the stream.) Data stream algorithms for frequency moments are well-studied and use sophisticated techniques based on hashing to produce high quality estimates. We show that REPEAT-RATE can be computed in one pass using space and update time of $O(\varepsilon^{-2}(\log n + \log m))$, matching the resources used by the well-known data stream algorithm for F_2 [Alon et al. 1999]. Our algorithm exploits some simple but powerful features of the classical data stream algorithm for F_2. We make the general observation that for any aggregator, an unbiased estimator for that aggregator over streams is also an unbiased estimator for that aggregator over probabilistic streams. Now, the basic estimator for F_2 simply hashes the stream and then applies a simple aggregration algorithm on the hashed input. We design a simple algorithm for this very same aggregator on probabilistic streams. Although this technique works in general, we have no *a priori* guarantee that estimators found in this way will have small variance in the probabilistic-stream setting. Still, we show that the variance associated with the estimator for F_2 is small by analyzing its algebraic structure. After demonstrating that the variance is small, we appeal to a standard set of techniques to get the estimate to within the desired relative error with high confidence.

We finish by considering the problems of answering OLAP-style queries involving *roll-ups* and *drill-downs* in a more efficient manner. Traditionally this is achieved by

keeping a small set of sufficient statistics for queries at lower levels of the dimension hierarchy so that they can be combined to answer queries at higher levels of the hierarchy. Ideally, the size of this set should be independent of the size of the database. For MEAN, we show that it suffices to keep a $O(\log \varepsilon^{-1})$-size set of sufficient statistics in order to answer such related queries. For F_2, it also suffices to keep a small set of sufficient statistics; for this case, the set-size is $O(\varepsilon^{-2})$. For MIN/MAX, we show that the algorithm in [Jayram et al. 2007] yields a set of $O(\varepsilon^{-1})$ sufficient statistics.

2. ESTIMATING MEAN

As has been noted previously in [Burdick et al. 2005a; Jayram et al. 2007], this is more technically challenging than might be expected. The naïve approach, simply computing SUM and COUNT and looking at their quotient, fails to be a good estimator in even simple examples. Consider the example from [Jayram et al. 2007], in which we have a probabilistic stream $\{\langle 1, 1.0\rangle\}, \{\langle 10, 0.1\rangle\}$. The true value of MEAN is 1.45, while the value of SUM/COUNT is approximately 1.81. Despite this, we will see that SUM/COUNT is a good approximation when COUNT is sufficiently large.

The difficulty in computing MEAN arises in part since we must divide by the number of items actually occurring in the stream, a quantity that varies over different possible worlds. The approach taken in [Jayram et al. 2007] was to rewrite the expression for the expected value of the average in terms of a generating function. Taking a derivative of this expression eliminated the division, but then required us to estimate an integral.

Their computation was based on the observation that only two relevant values were needed for each probabilistic item (i.e., sufficient statistics): p_i, the probability that item i is not $\perp$; and a_i, the expected value of item i, given that it is not $\perp$. They then show the following.

PROPOSITION 1 [JAYRAM ET AL. 2007]. MEAN *can be rewritten as*

$$\mathsf{MEAN} = \rho \int_0^1 \sum_{i \in [m]} a_i p_i \cdot \prod_{j \neq i} (1 - p_j + p_j x) \mathrm{d}x$$

where $\rho = 1/(1 - \prod_{i \in [m]}(1 - p_i))$.

We note that their definition of MEAN assumed that at least one item in the stream appeared with probability 1. That is, $\rho = 1$. Under our definition, we allow a stream to have no certain elements; this is equivalent to the formulation of [Jayram et al. 2007], divided by the probability that the stream contains at least one element (i.e., $1 - \prod_{i \in [m]}(1 - p_i)$). Hence, we have modified the original statement by multiplying by the proper factor. This does not qualitatively affect any of the arguments, and we do not belabor the point any further.

The approach of [Jayram et al. 2007] led to an $O(\log m)$-pass algorithm that estimated the integral; each pass required $O(\varepsilon^{-1} \log m)$ update time per item. Although this may be adequate for off-line applications, it is simply too slow for on-line applications with large m and small ε. Here, we provide a one-pass algorithm with update time $O(\log \varepsilon^{-1})$, an exponential improvement.

Our approach uses a careful, technically-involved analysis of the integral. We might hope that a simple Taylor-series expansion would give us a good estimate. Unfortunately, this approach fails. (To illustrate this, take $a_i = 1, p_i = 1/2$ for all i. Note that the integrand in Proposition 1 is only interesting near $x = 1$, while the coefficients of the Taylor series expansion about $x = 1$ grow exponentially.) However, by properly rewriting the integral, we get a step closer. Define

$$g(z) = \prod_{i\in[m]} (1 - p_i z), \quad h(z) = \sum_{i\in[m]} \frac{a_i p_i}{1 - p_i z}, \qquad \text{and} \quad f(z) = g(z)h(z) \ .$$

Notice that $f(z)$ is actually a polynomial, and further, $f(1-x)$ is precisely the integrand of Proposition 1. Applying a change of variable to the integral in Proposition 1 (setting $z = 1 - x$), and rewriting somewhat, we see that

$$\mathsf{MEAN} = \rho \int_0^1 g(z)h(z)\mathrm{dz} \ .$$

Not surprisingly, it will be easier to approximate g and h when z is bounded away from 1. Let $P = \sum_i p_i = \mathsf{COUNT}$, and for any $\varepsilon < 1/2$, define $z_0 = \min\{1, \frac{1}{P}\ln(2P/\varepsilon)\}$ if $P \geq 1$ and $z_0 = 1$ otherwise. We have the following lemma. (It is actually a special case of Lemma 5 appearing in Section 2.2.)

LEMMA 2. *Let $\varepsilon < 1/2$, with z_0 defined as above. Then*

$$(1 - \varepsilon)\mathsf{MEAN} \leq \rho \int_0^{z_0} g(z)h(z)\mathrm{dz} \leq \mathsf{MEAN} \ .$$

One of our key insights is that, whereas approximating the integrand directly is difficult, approximating the functions individually can be done with low-degree polynomials. A second key insight is that, while approximating $h(z)$ can be done directly with a Taylor-series expansion, we need to approximate the *logarithm* of $g(z)$. If z_0 is sufficiently small, say $z_0 \leq \theta$ for some $\theta < 1/2$, then the approximations of $\ln g(z)$ and $h(z)$ can be written as Taylor series with quickly converging remainder terms. However, if z_0 is near 1 (which happens when P is small), then we need an extra trick. In this case, we let $\theta \in [\varepsilon, 1/2]$, and define

$$I = \{i \in [m] : p_i \leq \theta\} \quad \text{and} \quad J = [m] - I \ .$$

We further define $g_I(z) = \prod_{i\in I}(1 - p_i z)$ and $h_I(z) = \sum_{i\in I} \frac{a_i p_i}{1-p_i z}$. Then we will see that the approximations of $\ln g_I(z)$ and $h_I(z)$ converge because the coefficients are tiny. Further, it is not hard to see that $|J|$ is necessarily small, since P must be small; in fact, $|J| \leq P/\theta$. So $f_J(z), g_J(z)$ are already low-degree polynomials; we simply calculate them. Amazingly, the degree of the approximating polynomials is independent of m.

With this approach, we can find good approximations to $g_I(z)$ and $h_I(z)$. But we are not quite done. Our approximation to $h_I(z)$ is a low-degree polynomial, while our approximation to $g_I(z)$ is exp(low-degree polynomial). There is no closed-form evaluation for the integral of the product of these two expressions. So we approximate our *approximation* of $g(z)$, as a polynomial. The details appear in Section 2.2.

In fact, when P is very large (and $I = [m]$), $h(z)$ is approximately SUM, while $\ln g(z)$ is approximately $-Pz$ (i.e. polynomials of degree 0 and 1, respectively). Applying this argument more carefully allows us to bound the error of using $\mathsf{SUM}/P = \mathsf{SUM}/\mathsf{COUNT}$ to approximate MEAN. We show this in Section 2.1.

Finally, we develop an entirely different approach for approximating the mean in Section 2.3. Although the bounds we obtain are not as strong, the techniques may be of independent interest.

2.1 Long Streams

In this subsection, we show that $\mathsf{SUM}/\mathsf{COUNT}$ is a good approximation to MEAN when $P = \mathsf{COUNT}$ is large. (Somewhat more accurately, we show $\rho \cdot \mathsf{SUM}/\mathsf{COUNT}$ is a good approximation.) The techniques we use here will serve as a warm-up to the more involved analysis of the next subsection. We have the following.

THEOREM 1. *For any $\delta \in [0,1)$ and for any $P > 0$, we have*

$$\rho \frac{\mathsf{SUM}}{\mathsf{COUNT}} \cdot \frac{\delta}{\ln(\frac{1}{1-\delta})}(1-(1-\delta)^P) \leq \mathsf{MEAN} \ .$$

Furthermore, for $P \geq e$,

$$\rho \frac{\mathsf{SUM}}{\mathsf{COUNT}} \left(1 - \frac{2\ln P}{P}\right) \leq \mathsf{MEAN} \leq \rho \frac{\mathsf{SUM}}{\mathsf{COUNT}} \left(1 + \frac{1}{P-1}\right) \ .$$

PROOF. We first prove the upper bound.

$$\mathsf{MEAN} = \rho \sum_{i\in[m]} a_i p_i \int_0^1 \prod_{j\neq i}(1-p_j z)\mathrm{dz} \leq \rho \sum_{i\in[m]} a_i p_i \int_0^1 e^{-z\sum_{j\neq i} p_j}\mathrm{dz} \ .$$

Now, for any i, we have

$$\int_0^1 e^{-z\sum_{j\neq i} p_j}\mathrm{dz} \leq \int_0^1 e^{-z(P-1)}\mathrm{dz} = \frac{e^{-z(P-1)}}{-(P-1)}\bigg|_0^1 \leq \frac{1}{P-1} \ .$$

Recalling that $P = \mathsf{COUNT}$, we have

$$\mathsf{MEAN} \leq \rho \sum_{i\in[m]} a_i p_i \frac{1}{P-1} = \rho \frac{\mathsf{SUM}}{P}\frac{P}{P-1} = \rho \frac{\mathsf{SUM}}{\mathsf{COUNT}} \left(1 + \frac{1}{P-1}\right) \ .$$

For the other inequality, let $\delta \in [0,1)$. We use the fact that $h(z) = \sum_{i\in[m]} a_i p_i (1-p_i z)^{-1} \geq \mathsf{SUM}$ for all $z \geq 0$. Further, noting that $1 - y \geq (1-\delta)^{y/\delta}$ for $y \in [0,\delta]$, we get that for $z \leq \delta$,

$$g(z) = \prod_{i\in[m]} (1-p_i z) \geq \prod_{i\in[m]} (1-\delta)^{p_i z/\delta} = (1-\delta)^{Pz/\delta} \ .$$

Putting this together, we have

$$\begin{aligned}\mathsf{MEAN} = \rho \int_0^1 g(z)h(z)\mathrm{dz} &\geq \rho \int_0^\delta (1-\delta)^{Pz/\delta} \cdot \mathsf{SUM}\mathrm{dz} \\ &= \rho \frac{\mathsf{SUM}}{\mathsf{COUNT}} \cdot \frac{\delta}{\ln(\frac{1}{1-\delta})}(1-(1-\delta)^P) \ ,\end{aligned}$$

proving the first inequality.

Examining the Taylor series reveals that $-\ln(1-\delta) = \sum_{i=1}^{\infty} \delta^i/i \leq \delta + \delta^2$ for $\delta \leq 1/2$. Hence,

$$\frac{\delta}{\ln(\frac{1}{1-\delta})} \geq \frac{1}{1+\delta} \geq 1-\delta \quad \text{for } \delta \leq 1/2.$$

So we have, for $\delta \leq 1/2$,

$$\mathsf{MEAN} \geq \rho \frac{\mathsf{SUM}}{\mathsf{COUNT}}(1-\delta)(1-(1-\delta)^P) \geq \rho \frac{\mathsf{SUM}}{\mathsf{COUNT}}(1-\delta)(1-e^{-\delta P}) \ .$$

Setting $\delta = \ln P/P$, and noting $\ln P/P \leq 1/e \leq 1/2$ for $P \geq e$, we have for $P \geq e$,

$$\mathsf{MEAN} \geq \rho \frac{\mathsf{SUM}}{\mathsf{COUNT}}(1 - 2\ln P/P) \ .$$

□

With some algebra, we have the following corollary.

COROLLARY 3. *For any $\varepsilon < 1$, if $P \geq \frac{4}{\varepsilon}\ln(2/\varepsilon)$, then*

$$(1-\varepsilon)\rho \frac{\mathsf{SUM}}{\mathsf{COUNT}} \leq \mathsf{MEAN} \leq (1+\varepsilon)\rho \frac{\mathsf{SUM}}{\mathsf{COUNT}}$$

PROOF. We first prove the lower bound. Since $P \geq \frac{4}{\varepsilon}\ln(2/\varepsilon)$, we have

$$\frac{2\ln P}{P} \leq \frac{2\ln(\frac{4}{\varepsilon}\ln(2/\varepsilon))}{\frac{4}{\varepsilon}\ln(2/\varepsilon)} = \frac{2(\ln 2 + \ln(\frac{2}{\varepsilon}) + \ln\ln(2/\varepsilon))}{\frac{4}{\varepsilon}\ln(2/\varepsilon)} < \frac{4\ln(2/\varepsilon)}{\frac{4}{\varepsilon}\ln(2/\varepsilon)} = \varepsilon \ ,$$

where the second line follows because $\ln 2 + \ln\ln(2/\varepsilon) < \ln(2/\varepsilon)$ for $\varepsilon < 1$.

For the upper bound, note that since $\varepsilon < 1$ and $P \geq \frac{4}{\varepsilon}\ln(2/\varepsilon)$, we have that $P > 2$. Hence, $P - 1 > P/2$. So we have

$$\frac{1}{P-1} < \frac{2}{P} \leq \frac{2}{\frac{4}{\varepsilon}\ln(2/\varepsilon)} < 2/(4/\varepsilon) = \varepsilon/2$$

The proof follows. □

Therefore, in the case that we know COUNT (i.e., P) is very large, or when we are not concerned with estimating MEAN extremely well, then the simple method of using SUM/COUNT works well. However, when COUNT is small or we need to be very accurate then using SUM/COUNT provides an inadequate estimate. We address this in the next subsection.

2.2 Short Streams

We now give an extremely efficient one-pass streaming algorithm that estimates MEAN within a multiplicative $(1+\varepsilon)$ factor. Remarkably, both the update time and the number of memory registers are independent of m, the number of items in the stream. It is also interesting that the algorithm is entirely deterministic, so it is guaranteed to return a good estimate. In contrast to typical streaming algorithms, there is no chance of failure.

We note that throughout this section, we may assume that $P < (4/\varepsilon)\ln(2/\varepsilon)$, since the last section showed that there is a simple algorithm for P greater than

this value. Let $\varepsilon < 1/2$, and $\theta \in [\varepsilon, 1/2]$. Our datastream algorithm maintains

$$\sum_{i\in[m]} p_i^k \quad \text{and} \quad \sum_{i\in[m]} a_i p_i^k \text{ for } k \in [O\left(\frac{\log(1/\varepsilon)}{\log(1/\theta)}\right)] \ ,$$

where the exact values for k are given below. In addition, we also remember the values a_j, p_j for every j such that $p_j > \theta$, until $P \geq (3/\theta)\ln(2/\varepsilon)$. (Notice that we will need to remember at most $(3/\theta^2)\ln(2/\varepsilon)$ such values.) Using these maintained values, we can quickly compute a $(1+5\varepsilon)$-approximations for MEAN. The full pseudocode for this algorithm is given in the appendix.

Somewhat more formally, let $I = \{i \in [m] : p_i \leq \theta\}$ if $P < (3/\theta)\ln(2/\varepsilon)$, and let $I = [m]$ otherwise. Let $J = [m] - I$, and define

$$P_k = \sum_{i\in I} p_i^k, \quad A_k = \sum_{i\in I} a_i p_i^k, \quad f_J(z) = \sum_{i\in J} a_i p_i \prod_{\substack{j\in J\\ j\neq i}} (1-p_j z), \quad g_J(z) = \prod_{j\in J}(1-p_j z) \ .$$

Note that given the maintained values, it is easy to calculate $f_J(z)$, $g_J(z)$ and P_k, A_k for $k = 1, 2, \ldots, O(\log(1/\varepsilon)/\log(1/\theta))$. We have our main result:

THEOREM 2. *Let $\varepsilon < 1/2$, $\theta \in [\varepsilon, 1/2]$, and define P_k, A_k, I, J, f_J, g_J as above and $\rho = 1/(1 - \prod_{i\in[m]}(1-p_i))$, and suppose that $P < \frac{4}{\varepsilon}\ln(2/\varepsilon)$. If $P \geq 1$, define ℓ to be the smallest even integer greater than or equal to $5\ln(2P/\varepsilon)$ and define $z_0 = \min\{1, \frac{1}{P}\ln(2P/\varepsilon)\}$; if $P < 1$, define ℓ to be the smallest even integer greater than or equal to $\max\{\ln(2/\varepsilon), 7\}$ and define $z_0 = 1$. Define*

$$\widetilde{\mathsf{MEAN}} = \frac{\rho}{1-\varepsilon}\int_0^{z_0} \prod_{i=1}^{k_0}\left(\sum_{j=0}^{\ell}\frac{(-P_i z^i)^j}{j!\cdot i^j}\right)\left(f_J(z) + g_J(z)\sum_{i=0}^{k_1} A_{i+1} z^i\right) \mathrm{dz}$$

where $k_0 = 2\ln(2/\varepsilon)/\ln(1/\theta)$, $k_1 = \ln(2/\varepsilon)/\ln(1/\theta)$. Then

$$\mathsf{MEAN} \leq \widetilde{\mathsf{MEAN}} \leq (1+5\varepsilon)\mathsf{MEAN} \ .$$

Before proving this main result, we first note an important corollary, which follows from Corollary 3 and Theorem 2.

COROLLARY 4. *Fix $\varepsilon < 1/2$ and $\theta \in [\varepsilon, 1/2]$. There is a single-pass $(\varepsilon, 0)$-approximation datastream algorithm for MEAN with update time $O(\ln(1/\varepsilon)/\ln(1/\theta))$ and using $O(\theta^{-2}\ln(1/\varepsilon) + \ln(1/\varepsilon)/\ln(1/\theta))$ registers[2]. In order to transform the maintained sketch into an estimate for MEAN, the algorithm takes an additional time of $O(\ln(1/\varepsilon)\cdot(\theta^{-2} + \ln^2(1/\varepsilon))\cdot(\ln\ln(1/\varepsilon) + \ln(1/\theta)))$. Here, we assume that arithmetic operations take $O(1)$ time, and probabilities and values are representable in $O(\log n)$ space. In particular,*

—*Setting $\theta = 1/e$ yields an $(\varepsilon, 0)$-approximation for MEAN with update time $O(\ln(1/\varepsilon))$, using $O(\ln(1/\varepsilon)\ln(mn))$ space, and with $O(\ln^3(1/\varepsilon)\ln\ln(1/\varepsilon))$ reconstruction time.*

—*Setting $\theta = \varepsilon^{1/2}$ yields an $(\varepsilon, 0)$-approximation for MEAN with update time $O(1)$, using $O(\frac{1}{\varepsilon}\ln(mn))$ space, and with reconstruction time of $O(\varepsilon^{-1}\ln^2(1/\varepsilon))$.*

[2]We assume a single register may store $O(\log m + \log n)$ bits

PROOF. The algorithm simply maintains the value

$$P_k^{big} = \sum_{i\in[m]} p_i^k \quad \text{and} \quad P_k^{small} = \sum_{i\in[m]:p_i\le\theta} p_i^k$$

for $k \in [k_0]$, where k_0 is defined as in Theorem 2. It also maintains

$$A_k^{big} = \sum_{i\in[m]} a_i p_i^k \quad \text{and} \quad A_k^{small} = \sum_{i\in[m]:p_i\le\theta} a_i p_i^k$$

for $k \in [k_1]$, where k_1 is as in Theorem 2 (hence, $k_0, k_1 \in O(\ln(1/\varepsilon)/\ln(1/\theta))$). We also maintain the value ρ. Finally, we store the values of a_i, p_i for all i such that $p_i > \theta$, until $P = P_1^{big} \ge (3/\theta)\ln(2/\varepsilon)$. If P ever grows larger than $(3/\theta)\ln(2/\varepsilon)$, then we may throw away the stored values of the a_i, p_i, as well as the value of P_k^{small} and A_k^{small} for all k. If P ever grows larger than $\frac{4}{\varepsilon}\ln(2/\varepsilon)$, then we may throw away all of the maintained values except for $P_1^{big} = \mathsf{SUM}$, $A_1^{big} = \mathsf{COUNT}$, and ρ. (Notice that if $P < (3/\theta)\ln(2/\varepsilon)$, then $P_k^{small} = P_k$ and $A_k^{small} = A_k$; otherwise $P_k^{big} = P_k$ and $A_k^{big} = A_k$. Notice additionally that maintaining P_k^{small}, P_k^{big} and the values p_i for all $p_i > \theta$ is somewhat redundant—and likewise for the A_k. However, we include it for clarity.)

Clearly, the update time for this algorithm is $O(k_0 + k_1 + 1) = O(\ln(1/\varepsilon))$. Furthermore, the number of i such that $p_i > \theta$ is at most P/θ. Since we stop storing these values once $P > (3/\theta)/\ln(2/\varepsilon)$, this takes at most $3\theta^{-2}\ln(2/\varepsilon)$ registers. Hence, the total number of registers is at most

$$O(2k_0 + 2k_1 + 3\theta^{-2}\ln(2/\varepsilon)) = O(\theta^{-2}\ln(1/\varepsilon) + \ln(1/\varepsilon)/\ln(1/\theta)) \ .$$

If $P \ge \frac{4}{\varepsilon}\ln(2/\varepsilon)$, the algorithm outputs the value $\rho/(1-\varepsilon)\cdot(\mathsf{SUM}/\mathsf{COUNT})$. By Corollary 3, this is a $(5\varepsilon, 0)$-approximation.

Otherwise, the algorithm outputs the value of $\widetilde{\mathsf{MEAN}}$ as defined in Theorem 2. Note that we have maintained sufficient information to obtain this value (and the value of z_0, which depends on P, is not needed until reconstruction time). The time to evaluate $\widetilde{\mathsf{MEAN}}$ is dominated by the time to multiply the various polynomials in the integrand; once this polynomial is found, integration is simple. But the total degree of the polynomial is at most

$$k_0^2\ell + k_1 + |J| \le k_0^2\ell + k_1 + 3\theta^{-2}\ln(2/\varepsilon) = O(\ln^3(1/\varepsilon) + \theta^{-2}\ln(1/\varepsilon)) \ .$$

Multiplying polynomials whose product has degree d can be done in $O(d\log d)$ time (see [Lipson 1981]). Hence, the reconstruction time follows.

Finally, the above proof provides a $(5\varepsilon, 0)$-approximation. Of course, applying the same technique using $\varepsilon/5$ yields the desired $(\varepsilon, 0)$-approximation. □

We now proceed with the proof of Theorem 2. In order to accommodate the fact that we may need to remember a_i, p_i for all i such that $p_i > \theta$, we need a slight generalization of Lemma 2. Recall that

$$g_I(z) = \prod_{i\in I}(1 - p_i z), \quad h_I(z) = \sum_{i\in I} \frac{a_i p_i}{1 - p_i z} \ .$$

Using the fact that $g(z) = g_I(z)g_J(z)$, that $h(z) = h_I(z) + h_J(z)$, and that $f_J(z) = g_J(z)h_J(z)$, we see

$$\begin{aligned}\mathsf{MEAN} &= \rho \int_0^1 g(z)h(z)\mathrm{dz} = \rho \int_0^1 g_I(z)g_J(z)(h_I(z) + h_J(z))\mathrm{dz} \\ &= \rho \int_0^1 g_I(z)(h_I(z)g_J(z) + f_J(z))\mathrm{dz} \ .\end{aligned}$$

The following lemma shows that the mass of the integrand is concentrated near $z = 0$.

LEMMA 5. *Let $\varepsilon < 1/2$. Further, define $z_0 = \min\{1, \frac{1}{P}\ln(2P/\varepsilon)\}$ for $P \geq 1$, and $z_0 = 1$ otherwise. Then for any $I \subseteq [n]$, $J = [n] - I$,*

$$(1-\varepsilon)\mathsf{MEAN} \leq \rho \int_0^{z_0} g_I(z)(h_I(z)g_J(z) + f_J(z))\mathrm{dz} \leq \mathsf{MEAN} \ .$$

PROOF. We may assume $z_0 = \frac{1}{P}\ln(2P/\varepsilon)$, for otherwise $z_0 = 1$ and the lemma follows trivially. Thus, we may also assume $P \geq 1$.

Notice that for any $I, J = [n] - I$,

$$g_I(z)(h_I(z)g_J(z) + f_J(z)) = g_I(z)g_J(z)(h_I(z) + h_J(z)) = f(z) \ .$$

Next, note that $f(z)$ is a decreasing function on $[0, 1]$:

$$\text{For all } i \in [n], \ \ \frac{\partial}{\partial z}\left(\prod_{j\in[n], j\neq i}(1 - p_j z)\right) < 0$$

Hence,

$$\frac{\partial}{\partial z} f = \frac{\partial}{\partial z}\left(\sum_i a_i p_i \prod_{j\neq i}(1 - p_j z)\right) = \sum_i a_i p_i \frac{\partial}{\partial z}\left(\prod_{j\in[n], j\neq i}(1 - p_j z)\right) < 0 \ .$$

So we have

$$\begin{aligned}\int_{z_0}^1 f(z)\mathrm{dz} &\leq f(z_0)(1 - z_0) = g(z_0)h(z_0)(1 - z_0) \\ &= \prod_i (1 - p_i z_0)\sum_i \frac{a_i p_i}{1 - p_i z_0}(1 - z_0) \\ &\leq e^{-Pz_0}\sum_i a_i p_i = \mathsf{SUM}\cdot e^{-Pz_0} \ .\end{aligned}$$

Recalling that $z_0 = \frac{\ln(2P/\varepsilon)}{P}$, we see that $e^{-Pz_0} = \frac{\varepsilon}{2P}$. Hence,

$$\rho \int_{z_0}^1 f(z)\mathrm{dz} \leq \rho\mathsf{SUM}\frac{\varepsilon}{2P} = \frac{\varepsilon}{2}\rho\frac{\mathsf{SUM}}{\mathsf{COUNT}} \ .$$

We now appeal to Theorem 1. Recall that

$$\mathsf{MEAN} \geq \rho\frac{\mathsf{SUM}}{\mathsf{COUNT}} \cdot \frac{\delta}{\ln(\frac{1}{1-\delta})}(1 - (1-\delta)^P)$$

for any $\delta \in [0,1)$. Since, by our assumption, $z_0 = \frac{1}{P}\ln(2P/\varepsilon) \le 1$, we see that $P \ge \ln(2/\varepsilon) > \ln 4$ since $\varepsilon < 1/2$. Using $P \ge \ln 4$ and $\delta = 1/2$ in the above inequality yields

$$\mathsf{MEAN} \ge \frac{\rho}{2}\frac{\mathsf{SUM}}{\mathsf{COUNT}} .$$

Hence, we see that

$$\rho\int_{z_0}^{1} f(z)\mathrm{dz} \le \frac{\varepsilon}{2}\rho\frac{\mathsf{SUM}}{\mathsf{COUNT}} \le \varepsilon\mathsf{MEAN} .$$

So we have

$$\mathsf{MEAN} = \rho\int_0^{z_0} f(z)\mathrm{dz} + \rho\int_{z_0}^{1} f(z)\mathrm{dz} \le \rho\int_0^{z_0} f(z)\mathrm{dz} + \varepsilon\mathsf{MEAN} .$$

Hence, $\rho\int_0^{z_0} f(z)\mathrm{dz} \ge (1-\varepsilon)\mathsf{MEAN}$. The other side of the inequality follows trivially since $z_0 \le 1$. □

Define

$$\widetilde{g}_k(z) = \exp\left(-\sum_{j=1}^{k}\frac{P_j}{j}z^j\right) \quad \text{and} \quad \widetilde{h}_k(z) = \sum_{j=0}^{k} A_{j+1}z^j .$$

We will show that these are good approximations to g_I, h_I, respectively. The key idea is that for large P, z_0 is small, while for small P, P_i and A_i decrease as θ^i. More specifically, we have the following claim.

Claim 6. *For all $i > 0$ and $0 \le z \le z_0$, we have $P_i z^i \le (Pz)\theta^{i-1}$ and $A_{i+1}z^i \le A_1\theta^i$.*

Proof. First, note that $(c + \ln x)/x$ is a decreasing function for $x \ge e$ and constant $c > 0$. Hence, when $P \ge (3/\theta)\ln(2/\varepsilon) > e$, we have

$$z_0 \le \frac{\ln(2P/\varepsilon)}{P} \le \frac{\ln(2/\varepsilon) + \ln((3/\theta)\ln(2/\varepsilon))}{(3/\theta)\ln(2/\varepsilon)} = \frac{\ln(2/\varepsilon) + \ln(2/\varepsilon) + \ln(\frac{3}{2}\frac{\varepsilon}{\theta}\ln(2/\varepsilon))}{(3/\theta)\ln(2/\varepsilon)}$$
$$\le \frac{3\ln(2/\varepsilon)}{(3/\theta)\ln(2/\varepsilon)} = \theta ,$$

where the last line follows since $\frac{3}{2}\frac{\varepsilon}{\theta} \le \frac{3}{2} < 2/\varepsilon$ for $\varepsilon < 1/2$ and $\theta \ge \varepsilon$. So in the case that $P \ge (3/\theta)\ln(2/\varepsilon)$, we have (by definition) $J = \emptyset$, and $P_k z^k = \sum_{i\in[n]} p_i^k z^k \le Pz\theta^{k-1}$ for $z \le z_0$.

On the other hand, if $P < (3/\theta)\ln(2/\varepsilon)$, then I contains only indices i such that $p_i \le \theta$. Hence, $P_k z^k = \sum_{i\in I} p_i^k z^k \le \sum_{i\in I} p_i^k z \le Pz\theta^{k-1}$ for $z \le z_0 \le 1$.

So in both cases, $P_k z^k \le Pz\theta^{k-1}$. A similar argument shows that $A_{k+1}z^k \le A_1\theta^k$ for $z \le z_0$. □

With the claim in hand, we are ready to show that $\widetilde{g}_k$ and $\widetilde{h}_k$ are good approximations.

Lemma 7. *Define $\widetilde{g}_{k_0}$ and $\widetilde{h}_{k_1}$ as above, and let $k_0 \ge 2\ln(2/\varepsilon)/\ln(1/\theta)$ and $k_1 \ge \ln(2/\varepsilon)/\ln(1/\theta)$. Then*

$$g_I(z) \le \widetilde{g}_{k_0}(z) \le (1+\varepsilon)g_I(z) \quad \text{and} \quad h_I(z) \le \widetilde{h}_{k_1}(z) \le (1+\varepsilon)h_I(z) .$$

PROOF. We first do a Taylor series expansion for the *logarithm* of $g_I(z)$. (The series is absolutely convergent for $z \in [0,1)$, so we do not go through the formality of writing the above expression as a finite sum with a remainder term that goes to 0.)

$$\ln g_I(z) = \sum_{i\in I} \ln(1-p_i z) = -\sum_{i\in I}\sum_{j\geq 1} \frac{p_i^j z^j}{j} = -\sum_{j\geq 1} \frac{P_j}{j} z^j \ .$$

Hence, using Claim 6, we see

$$0 \leq \ln \widetilde{g}_{k_0}(z) - \ln g_I(z) \ \leq \sum_{j\geq k_0+1} \frac{P_j z^j}{j} \leq \frac{Pz_0}{k_0+1} \sum_{j\geq k_0+1} \theta^{j-1} \leq \frac{Pz_0}{k_0+1} 2\theta^{k_0}$$

since $\theta \leq 1/2$. For $k_0 \geq (\ln(2/\varepsilon) + \ln(Pz))/\ln(1/\theta)$ (assuming $k_0 \geq 1$), this bounds the error by $\varepsilon/2$. But if $P \geq 1$, the value of Pz_0 is bounded above by $\ln(2P/\varepsilon)$, while otherwise, $Pz_0 < 1$. Hence, the error is bounded by $\varepsilon/2$ so long as

$$k_0 \geq (\ln(2/\varepsilon) + \ln(\max\{\ln(2P/\varepsilon), 1\}))/\ln(1/\theta) \ .$$

Since $2/\varepsilon > \max\{\ln(2P/\varepsilon), 1\}$ for $P \leq 4\varepsilon^{-1}\ln(2/\varepsilon)$ and $\varepsilon < 1/2$, this implies that the error is still bounded by $\varepsilon/2$ for $k_0 \geq 2\ln(2/\varepsilon)/\ln(1/\theta)$. Thus,

$$g_I(z) \leq \widetilde{g}_{k_0}(z) \leq g_I(z) \cdot \exp(\varepsilon/2) \leq g_I(z)(1 + (e-1)\varepsilon/2)) \leq g_I(z)(1+\varepsilon) \ .$$

where the second line follows from the fact that $e^y \leq 1 + (e-1)y$ for all $y \in [0,1]$.

We now approximate $h_I(z)$ using a Taylor series expansion. We have

$$h_I(z) \ = \ \sum_{i\in I} \frac{a_i p_i}{1-p_i z} = \sum_{i\in I} a_i p_i \sum_{j\geq 0} p_i^j z^j = \sum_{j\geq 0} A_{j+1} z^j \ .$$

Hence,

$$0 \leq h_I(z) - \widetilde{h}_{k_1}(z) \ \leq \sum_{j\geq k_1+1} A_{j+1} z^j \leq \sum_{j\geq k_1+1} A_1 \theta^j \leq h_I(z) \cdot \theta^{k_1}$$

since $\theta \leq 1/2$. Hence, for $k_1 \geq \ln(2/\varepsilon)/\ln(1/\theta)$,

$$h_I(z) \leq \widetilde{h}_{k_1}(z) \leq (1+\varepsilon) h_I(z) \ .$$

□

Thus we have reduced the problem to that of estimating

$$\int_0^{z_0} \widetilde{g}_{k_0}(z)(f_J(z) + g_J(z)\widetilde{h}_{k_1}(z))\mathrm{dz}$$

where $k_0 = 2\ln(2/\varepsilon)/\ln(1/\theta)$ and $k_1 = \ln(2/\varepsilon)/\ln(1/\theta)$. Unfortunately, $\widetilde{g}_{k_0}$ is not a polynomial, hence integrating is difficult. So we now expand it.

LEMMA 8. *Let $\varepsilon < 1/2$. If $P \geq 1$, let ℓ be the smallest even integer greater than $5\ln(2P/\varepsilon)$; otherwise, let ℓ be the smallest even integer greater than $\max\{\ln(2/\varepsilon), 7\}$. Then, for $z \leq z_0$,*

$$\widetilde{g}_{k_0}(z) \leq \prod_{i=1}^{k_0} \left(\sum_{j=0}^{\ell} \frac{1}{j!} \frac{(-P_i z^i)^j}{i^j} \right) \leq (1+\varepsilon)\widetilde{g}_{k_0}(z) \ .$$

PROOF. Recall that $\widetilde{g}_k(z) = \prod_{i=1}^{k} \exp\left(\frac{P_i}{i} z^i\right)$. We expand $\exp(-P_i z^i/i)$ as

$$\exp(-P_i z^i/i) = \sum_{j=0}^{\ell} \frac{1}{j!} \frac{(-P_i z^i)^j}{i^j} + R_i(z) \ ,$$

where $R_i(z) = \sum_{j \geq \ell+1} \frac{1}{j!} \frac{(-P_i z^i)^j}{i^j}$. We will first bound $R_i(z)$.

First, consider the case that $P \geq 1$. From Claim 6, we note that $P_i z^i \leq (Pz)\theta^{i-1} \leq \ln(2P/\varepsilon)\theta^{i-1}$. Consider the absolute value for an individual term in the series for $R_i(z)$:

$$\frac{1}{j!} \frac{(P_i z^i)^j}{i^j} \leq \left(\frac{e}{j}\right)^j \left(\frac{\ln(2P/\varepsilon)\theta^{i-1}}{i}\right)^j = \left(\frac{e \ln(2P/\varepsilon)}{j}\right)^j \left(\frac{\theta^{i-1}}{i}\right)^j \ .$$

Since $j \geq \ell + 1 > 5\ln(2P/\varepsilon)$, we see that (the absolute values of) these terms are decreasing (in j). Since the series is alternating and ℓ is even, we see that $R_i(z)$ is bounded by its $j = \ell + 1$ term:

$$0 \leq -R_i(z) \leq \frac{-1}{(\ell+1)!} \frac{(-P_i z^i)^{\ell+1}}{i^{\ell+1}} \leq \left(\frac{e \ln(2P/\varepsilon)}{\ell+1}\right)^{\ell+1} \left(\frac{\theta^{i-1}}{i}\right)^{\ell+1} \ .$$

Since $\ell > 5\ln(2P/\varepsilon)$ and $\theta \leq 1/2$, we have

$$\begin{aligned} -R_i(z) \ &\leq \ \left(\frac{e \ln(2P/\varepsilon)}{5\ln(2P/\varepsilon)}\right)^{5\ln(2P/\varepsilon)+1} \left(\frac{1}{i2^{i-1}}\right) \\ &\leq \ (e/5)^{5\ln(2P/\varepsilon)} \frac{1}{i2^{i-1}} < e^{-2\ln(2P/\varepsilon)} \left(\frac{1}{i2^{i-1}}\right) \\ &= \ \left(\frac{\varepsilon}{2P}\right)^2 \left(\frac{1}{i2^{i-1}}\right) < \left(\frac{1}{i2^i}\right) \varepsilon \cdot \left(\frac{\varepsilon}{2P}\right) \ . \end{aligned}$$

Note that since $z_0 \leq \ln(2P/\varepsilon)/P$, we have that $Pz_0 \leq \ln(2P/\varepsilon)$, hence $e^{-Pz_0} \geq \varepsilon/(2P)$. Furthermore, we have that $P_i z^i \leq P z_0 \theta^{i-1} < P z_0$ for $z \leq z_0$. So $e^{-Pz_0} \leq e^{-P_i z^i/i}$. Putting this together, we see that

$$-R_i(z) < \left(\frac{1}{i2^i}\right) \varepsilon \cdot \left(\frac{\varepsilon}{2P}\right) < \left(\frac{1}{i2^i}\right) \varepsilon \cdot e^{-Pz_0} \leq \left(\frac{1}{i2^i}\right) \varepsilon \cdot e^{-P_i z^i/i} \ .$$

Now, consider the case $P < 1$. Note that the absolute values of the terms in the series for $R_i(z)$ are again decreasing: in this case, is follows simply because $P_i z^i/i < 1$. So, again, since the series is alternating and ℓ is even, we may bound

$R_i(z)$:

$$\begin{aligned}
0 \leq -R_i(z) \leq \frac{-1}{(\ell+1)!}\frac{(-P_i z^i)^{\ell+1}}{i^{\ell+1}} &\leq \left(\frac{e}{\ell+1}\right)^{\ell+1}\left(\frac{(Pz)\theta^{i-1}}{i}\right)^{\ell+1} \\
&\leq \left(\frac{e}{\ell+1}\right)^{\ell+1}\left(\frac{1}{i2^{i-1}}\right) \\
&\leq \left(\frac{e}{9}\right)^{\ell+1}\left(\frac{1}{i2^{i-1}}\right) \\
&\leq \left(\frac{1}{e}\right)^{\ln(2/\varepsilon)+1}\left(\frac{1}{i2^{i-1}}\right) \\
&\leq \left(\frac{1}{e}\right)\left(\frac{1}{i2^{i}}\right)\varepsilon
\end{aligned}$$

where the fifth inequality follows since $\ell \geq 8$ and the sixth inequality follows since $\ell \geq \ln(1/\varepsilon)$ and $e/9 < 1/e$. In this case, note that $P_i z^i/i < 1$, hence $e^{-P_i z^i/i} > 1/e$. Thus, $-R_i(z) < \left(\frac{1}{i2^i}\right)\varepsilon \cdot e^{-P_i z^i/i}$.

So in both cases, we see that

$$0 \leq -R_i(z) \leq \left(\frac{1}{i2^i}\right)\varepsilon \cdot e^{-P_i z^i/i} \ .$$

We now bound our expression. We have

$$\prod_{i=1}^{k_0}\left(\sum_{j=0}^{\ell}\frac{1}{j!}\frac{(-P_i z^i)^j}{i^j}\right) = \prod_{i=1}^{k_0}\left(e^{-P_i z^i/i} - R_i(z)\right) \geq \prod_{i=1}^{k_0}\left(e^{-P_i z^i/i}\right) = \widetilde{g}_{k_0}(z)$$

since $R_i(z) \leq 0$. To bound the other side, we combine our bound for $-R_i(z)$ obtained above with the following claim, whose proof appears in the appendix.

Claim 9. *For all $k > 0$ and for $\varepsilon < 1/2$,*

$$\prod_{i=1}^{k}(1 + \frac{1}{i2^i}\varepsilon) \leq (1 + \frac{k}{k+1}\varepsilon) \ .$$

Given this, we have

$$\begin{aligned}
\prod_{i=1}^{k_0}\left(\sum_{j=0}^{\ell}\frac{1}{j!}\frac{(-P_i z^i)^j}{i^j}\right) &= \prod_{i=1}^{k_0}\left(e^{-P_i z^i/i} - R_i(z)\right) \\
&\leq \prod_{i=1}^{k_0}\left(e^{-P_i z^i/i}\left(1 + \frac{1}{i2^i}\varepsilon\right)\right) \\
&= \prod_{i=1}^{k_0}\left(e^{-P_i z^i/i}\right)\prod_{i=1}^{k_0}\left(1 + \frac{1}{i2^i}\varepsilon\right) \leq (1+\varepsilon)\widetilde{g}_{k_0}(z) \ .
\end{aligned}$$

□

To complete the proof of the theorem, we combine our lemmas:

$$\begin{aligned}
\widetilde{\mathsf{MEAN}} &= \frac{\rho}{1-\varepsilon}\int_0^{z_0}\prod_{i=1}^{k_0}\left(\sum_{j=0}^{\ell}\frac{(-P_i z^i)^j}{j!\cdot i^j}\right)\left(f_J(z)+g_J(z)\sum_{i=0}^{k_1}A_{i+1}z^i\right)\mathrm{dz}\\
&\le (1+\varepsilon)\frac{\rho}{1-\varepsilon}\int_0^{z_0}\widetilde{g}_{k_0}(z)\Big(f_J(z)+g_J(z)\widetilde{h}_{k_1}(z)\Big)\,\mathrm{dz} \quad \text{by Lemma 8}\\
&\le \frac{(1+\varepsilon)^3}{1-\varepsilon}\rho\int_0^{z_0} g_I(z)(f_J(z)+g_J(z)h_I(z))\,\mathrm{dz} \quad \text{by Lemma 7}\\
&\le \frac{(1+\varepsilon)^3}{1-\varepsilon}\mathsf{MEAN} \quad \text{by Lemma 5}\\
&\le (1+5\varepsilon)\mathsf{MEAN}\ .
\end{aligned}$$

The other side follows similarly:

$$\begin{aligned}
\widetilde{\mathsf{MEAN}} &= \frac{\rho}{1-\varepsilon}\int_0^{z_0}\prod_{i=1}^{k_0}\left(\sum_{j=0}^{\ell}\frac{(-P_i z^i)^j}{j!\cdot i^j}\right)\left(f_J(z)+g_J(z)\sum_{i=0}^{k_1}A_{i+1}z^i\right)\mathrm{dz}\\
&\ge \frac{\rho}{1-\varepsilon}\int_0^{z_0}\widetilde{g}_{k_0}(z)\Big(f_J(z)+g_J(z)\widetilde{h}_{k_1}(z)\Big)\,\mathrm{dz} \quad \text{by Lemma 8}\\
&\ge \frac{\rho}{1-\varepsilon}\int_0^{z_0} g_I(z)(f_J(z)+g_J(z)h_I(z))\,\mathrm{dz} \quad \text{by Lemma 7}\\
&\ge \mathsf{MEAN} \quad \text{by Lemma 5}\ .
\end{aligned}$$

2.3 Short Streams (Alternative Approach)

In this section, we briefly discuss an alternative idea in the short stream case that may be of independent interest, although the space and time bounds are not as good as those of the algorithm in the previous section. For the duration of this section let Y and Z be the random variables defined by,

$$Y = \sum_{i:X_i\neq\perp} X_i \quad \text{and} \quad Z = |\{i\in[m] : X_i \neq \perp\}|\ .$$

The main idea behind our algorithm is that if COUNT is not large then it is sufficient to estimate MEAN from $\Pr[Z=z]$ and $\mathbb{E}[Y|Z=z]$ for a relatively small range of values of z.

THEOREM 3. *There is a single-pass $(\varepsilon,0)$-approximation of* MEAN *that uses $O(\varepsilon^{-1/2}(\log n+\log m+\log\varepsilon^{-1}))$ space.*

PROOF. Let $c = 9\varepsilon^{-1}\ln(mn\varepsilon^{-1})$. First, if COUNT $\ge c$ then by Corollary 3, SUM/COUNT is an $(\varepsilon,0)$-approximation of MEAN. Alternatively, assume that COUNT $\le c$.

For $j\in[m]$, let

$$Z_j = |\{i\in[j] : X_i\neq\perp\}| \quad \text{and} \quad Y_j = \sum_{i\le j:X_i\neq\perp} X_i\ .$$

Algorithm 1 Stream Algorithm for MEAN

Takes, ε, θ as input, with $\theta \in [\varepsilon, 1/2]$, $\varepsilon < 1/2$.

Initialization:

Set $k_0 := 2\ln(2/\varepsilon)/\ln(1/\theta)$ and $k_1 := \ln(2/\varepsilon)/\ln(1/\theta)$.
Initialize $P_k^{big} = P_k^{small} := 0$ for all $k \in [k_0]$.
Initialize $A_k^{big} = A_k^{small} := 0$ for all $k \in [k_1]$.
Initialize $t := 0$. *// t tracks the number of indices in J that we have seen so far.*
Initialize $\alpha = 1$ *// We will maintain $\alpha = \prod_i (1 - p_i)$.*

Update: Item (a_i, p_i) arrives.

Update $\alpha := \alpha \cdot (1 - p_i)$.
Increment $P_k^{big} := P_k^{big} + p_i^k$ for all $k \in [k_0]$.
Increment $A_k^{big} := A_k^{big} + a_i p_i^k$ for all $k \in [k_1]$.
if $P_1^{big} < (3/\theta)\ln(2/\varepsilon)$, **then**:
  if $p_i > \theta$, **then**:
    Increment t by one.
    Store the values of $b_t := a_i$ and $q_t := p_i$.
  else
    Increment $P_k^{small} := P_k^{small} + p_i^k$ for all $k \in [k_0]$.
    Increment $A_k^{small} := A_k^{small} + a_i p_i^k$ for all $k \in [k_1]$.
  endif
endif

Reconstruction:

Set $\rho := 1/(1 - \alpha)$.
Set $z_0 := \min\{1, \frac{1}{P}\ln(2P/\varepsilon)\}$ if $P_1^{big} \geq 1$; otherwise, set $z_0 = 1$.
Set $\ell := 2\lceil(5/2)\ln(2P/\varepsilon)\rceil$ if $P_1^{big} \geq 1$; otherwise, set $\ell := 2\lceil\max\{\frac{1}{2}\ln(2/\varepsilon), 7/2\}\rceil$.
if $P_1^{big} \geq \frac{4}{\varepsilon}\ln(2/\varepsilon)$, **then**:
  return $\rho A_1^{big}/P_1^{big}$.
endif
if $P_1^{big} \geq \frac{3}{\theta}\ln(2/\varepsilon)$, **then**:
  Calculate $Q(z) = \frac{\rho}{1-\varepsilon}\prod_{i=1}^{k_0}\left(\sum_{j=0}^{\ell}\frac{(-P_i^{big}z^i)^j}{j!\cdot i^j}\right)\left(\sum_{i=0}^{k_1}A_{i+1}^{big}z^i\right)$
else
  Calculate $Q(z) = \frac{\rho}{1-\varepsilon}\prod_{i=1}^{k_0}\left(\sum_{j=0}^{\ell}\frac{(-P_i^{small}z^i)^j}{j!\cdot i^j}\right)$
    $\cdot\left(\sum_{i=1}^{t} b_i q_i \prod_{\substack{j\in[t]\\ j\neq i}}(1 - q_j z) + \prod_{j\in[t]}(1 - q_j z)\sum_{i=0}^{k_1}A_{i+1}^{small}z^i\right)$
endif
Let Q_i denote the coefficient for z^i in $Q(z)$.
Initialize *estimate* $:= 0$. *// We now integrate $Q(z)$ from 0 to z_0.*
for $i := 0$ to *degree*(Q):
  Increment *estimate* $:=$ *estimate* $+ Q_i z_0^{i+1}/(i+1)$.
endfor
return *estimate*.

Note that $\mathbb{E}[Y_m] = \mathsf{SUM}$ and $\mathbb{E}[Z_m] = \mathsf{COUNT}$. For $j, z \in [m]$, let

$$A_{j,z} = \Pr[Z_j = z] \quad \text{and} \quad B_{j,z} = \sum_y y \Pr[Y_j = y, Z_j = z] \ .$$

First note that for $j, z \in [m]$,

$$A_{j,z} = A_{j-1,z-1}p_j + A_{j-1,z}(1 - p_j)$$

where $A_{0,z} = 1$ if $z = 0$ and 0 otherwise. Similarly,

$$\begin{aligned}
B_{j,z} &= \sum_y y\big(\Pr[Y_{j-1} = y, Z_{j-1} = z, X_j = \perp] \\
&\qquad + \sum_a \Pr[Y_{j-1} = y - a, Z_{j-1} = z - 1, X_j = a] \big) \\
&= (1 - p_j)B_{j-1,z} + p_j \sum_{a,y} \Pr[X_j = a | X_j \neq \perp] \, y \Pr[Y_{j-1} = y - a, Z_{j-1} = z - 1] \\
&= (1 - p_j)B_{j-1,z} + p_j \sum_a \Pr[X_j = a | X_j \neq \perp] \, (a \Pr[Z_{j-1} = z - 1] + B_{j-1,z-1}) \\
&= (1 - p_j)B_{j-1,z} + p_j (\mathbb{E}[X_j | X_j \neq \perp] \, A_{j-1,z-1} + B_{j-1,z-1})
\end{aligned}$$

and $B_{0,z} = 0$ for all z. Hence, given $A_{j-1,z}$ and $B_{j-1,z}$ for all z, it is possible to compute $A_{j,z}$ and $B_{j,z}$ on seeing the ϑ_j.

In the rest of the proof we argue that ignoring any values of $A_{j,z}$ or $B_{j,z}$ that are smaller than some carefully chosen value β ensures that a) there are at most $O(\sqrt{\varepsilon}c)$ values of z such that $A_{j,z}$ or $B_{j,z}$ are non-zero and b) the error in the final answer is sufficiently small.

First note that if each of the above calculations is performed with additive error β, a simple induction argument shows that $A_{j,z}$ is computed with error,

$$(p_j + 1 - p_j)(j - 1)\beta + \beta = j\beta \ .$$

Similarly, $B_{j,z}$ is computed with error at most,

$$(1 - p_j)n(j - 1)^2\beta + p_j(n(j - 1)\beta + n(j - 1)^2\beta) + \beta \leq j^2 n\beta \ .$$

Therefore $\beta = \varepsilon m^{-2} n^{-2}$ suffices to compute

$$\mathsf{MEAN} = \sum_{y,z} \Pr[Y = y, Z = z] \, y/z = \sum_z B_{m,z}/z$$

with additive-error at most ε which translates into a relative-error since the quantity being estimated is $\Omega(1)$.

Secondly, note that because $\mathbb{E}[Z_j] < c$,

$$\Pr\left[|Z_j - \mathbb{E}[Z_j]| \geq \sqrt{\varepsilon}c\right] \leq 2\exp(\varepsilon c/3) < \beta/n \ ,$$

by an application of the Chernoff-Hoeffding bound. Hence, for each $j \in [m]$, there at most $O(\sqrt{\varepsilon}c)$ values of z such that $A_{j,z}$ or $B_{j,z}$ (noting $B_{j,z} \leq nA_{j,z}$) are greater than β. $\square$

3. ESTIMATING DISTINCT ITEMS

In this section, we present an (ε, δ)-approximation algorithm for DISTINCT, i.e., the expected value of F_0. Even with deterministic streams, approximating DISTINCT is nontrivial. Of course, with no memory limitations, the problem becomes much easier. But when space is bounded, the solution requires more work. A now standard deterministic stream technique from [Flajolet and Martin 1985] approaches the problem in the following way: Each item is hashed to a random value between 0 and 1 (so items with identical values will always be hashed to the same value). We track only the minimum hashed value. Intuitively, we expect that if this minimum value is $1/k$, then there are about $k-1$ distinct values. More accurate algorithms, such as that found in [Bar-Yossef et al. 2002], build on this basic idea.

A major part of our algorithm, in contrast to the algorithm from MEAN, is to actually randomly instantiate a deterministic stream. However, this approach will only give an (ε, δ)-approximation in small space if the expected number of distinct values is not very small. In the following theorem, we show that it is possible to also deal with this case. Furthermore, the random instantiation will be performed in a slightly non-obvious fashion for the purpose of achieving a tight analysis of the appropriate probability bounds.

THEOREM 4. *We can (ε, δ)-approximate* DISTINCT *in a single pass and taking $O((\varepsilon^{-2} + \log n) \log \delta^{-1})$ space.*

PROOF. First note that,

$$\mathsf{DISTINCT} = \sum_{j \in [n]} \left(1 - \prod_{i \in [m]} (1 - p_i(j)) \right) .$$

Consider $\mathsf{COUNT} = \sum_{i \in [m], j \in [n]} p_i(j)$. Then $\mathsf{DISTINCT} \leq \mathsf{COUNT}$ by the union bound and

$$e^{-\mathsf{COUNT}} \mathsf{COUNT} \leq 1 - e^{-\mathsf{COUNT}} \leq 1 - \prod_{i \in [m]} p_i(\perp) \leq \mathsf{DISTINCT} .$$

(The first inequality follows because $e^{-0}0 = 1 - e^{-0}$ and the derivative of $e^{-x}x$ is less than the derivative of $1 - e^{-x}$ for $x > 0$.) Hence if $\mathsf{COUNT} \leq \ln(1+\varepsilon)$ then COUNT is an $(\varepsilon, 0)$-approximation for DISTINCT. We now assume that $\mathsf{COUNT} > \ln(1+\varepsilon)$ and hence

$$\mathsf{DISTINCT} \geq \ln(1+\varepsilon) e^{-\ln(1+\varepsilon)} \geq \varepsilon/2$$

assuming ε is sufficiently small.

Let $c_1 = 3^3 \cdot 2\varepsilon^{-3} \ln(4/\delta)$ and consider the following algorithm:

(1) For $k \in [c_1]$ and each tuple $(j, p_i(j))$ in ϑ_i: place $jc_1 + k$ in an induced stream A' with probability $p_i(j)$.
(2) Compute an $(\varepsilon/3, \delta/2)$-approximation X of $F_0(A')/c_1$ using the algorithm of [Bar-Yossef et al. 2002].

CLAIM 10. $\Pr[|Z - \mathsf{DISTINCT}| \leq \varepsilon \mathsf{DISTINCT}/3] \leq \delta/2$ *where* $Z = F_0(A')/c_1$.

PROOF. By linearity of expectation $\mathbb{E}[Z] = \mathsf{DISTINCT}$. Furthermore, note that $c_1 Z$ can be thought of as the sum of nc_1 independent boolean trials corresponding to whether each $jc_1 + k$ appears in A' (some trials may have zero probability of success) and hence,

$$\Pr\left[\frac{|Z - \mathsf{DISTINCT}|}{\mathsf{DISTINCT}} \leq \frac{\varepsilon}{3}\right] \leq 2\exp\left(\frac{-\varepsilon^2 c_1 \mathsf{DISTINCT}}{27}\right) \leq 2\exp\left(\frac{-\varepsilon^3 c_1}{2 \cdot 27}\right) = \frac{\delta}{2} \ .$$

□

The result follows because X is a $(\varepsilon/3, \delta/2)$-approximation of $F_0(A')/c_1$. The space bound follows since all that is required the $O((\varepsilon^{-2} + \log n)\log \delta^{-1})$ space used by the algorithm of [Bar-Yossef et al. 2002].

4. ESTIMATING REPEAT-RATE

In this section, we address the problem of computing frequency moments on probabilistic data, with a particular application to the second frequency moment. For an input $\vec{a} = (a_1, a_2, \ldots, a_m)$ over a domain $[n]$, let f_a denote the frequency of element a in $\vec{a}$. We define $F_k(\vec{a}) = \sum_a f_a^k$. We write the expected value of F_2 as REPEAT-RATE.

We assume that each item in the data stream, ϑ_i, is of the form $\{(a_i, p_i)\}$. Let $(\vec{a}, \vec{p})$ denote a probabilistic stream where $\vec{a} = (a_1, a_2, \ldots, a_m)$ and $\vec{p} = (p_1, p_2, \ldots, p_m)$. Let h be a function (chosen randomly according to some distribution), and let $h(\vec{a})$ denote the vector obtained by applying h to each element in $\vec{a}$. The following simple lemma shows that any unbiased estimator working over streams is also an unbiased estimator working over probabilistic streams.

LEMMA 11. *Let Q and Q' be two aggregate functions such that $Q(\vec{b}) = \mathbb{E}_h[Q'(h(\vec{b}))]$ for any input $\vec{b}$ over the base domain. Let $Q(\vec{a}, \vec{p})$ denote the expected value of $Q(\vec{b})$ where $\vec{b}$ is a deterministic stream chosen according to the distribution specified by $(\vec{a}, \vec{p})$. Similarly for $Q'(\vec{a}, \vec{p})$. Then, for any probabilistic stream $(\vec{a}, \vec{p})$,*

$$Q(\vec{a}, \vec{p}) = \mathbb{E}_h[Q'(h(\vec{a}), \vec{p})] \ ,$$

i.e., $Q'(h(\vec{a}), \vec{p})$ is an unbiased estimator of $Q(\vec{a}, \vec{p})$.

PROOF. For any possible world, let $\vec{b} = (b_1, \ldots, b_m)$ denote the subset of $\vec{a}$ that appears in that world. The premise of the lemma implies that $Q(\vec{b}) = \mathbb{E}_h[Q(h(\vec{b}))]$. Taking the expected value over possible worlds, and interchanging the expectation on the right hand side yields the lemma. □

Thus, given an unbiased estimator for streams, our task reduces to finding an algorithm that efficiently computes the estimator over probabilistic streams. In fact, this technique is useful in deriving one-pass probabilistic stream algorithms for a variety of frequency moments. Although these algorithms have approximately the correct expectations, showing that they are tightly concentrated about their expected values must be done on an algorithm-by-algorithm basis. Here, we apply the method to approximating REPEAT-RATE.

Let us first briefly review the classical algorithm for F_2 over streams [Alon et al. 1999]. Let $\mathcal{H}$ denote a uniform family of 4-wise independent hash functions such

that $h : [n] \to \{-1, +1\}$ for each $h \in \mathcal{H}$. In other words, $h(z_1)$, $h(z_2)$, $h(z_3)$, and $h(z_4)$) are independent over the random choice of h for any distinct z_1, z_2, z_3, and z_4. For an input $\vec{a} = (a_1, a_2, \ldots, a_m)$, and a hash function $h \in \mathcal{H}$, define the estimator $B_h = (\sum_i h(a_i))^2$. Note that this can be computed in one pass. It is well-known that $\mathbb{E}_h[B_h] = F_2(\vec{a})$ and that $\mathbb{V}\mathrm{ar}_h[B_h] \leq 2F_2(\vec{a})^2$. Let the aggregator $\mathrm{Sum2}(b_1, b_2, \ldots)$ denote square of the sum of the input values, $(\sum_i b_i)^2$.

THEOREM 5. *For a probabilistic stream $(\vec{a}, \vec{p})$, the quantity $\mathrm{Sum2}(h(\vec{a}), \vec{p})$ is an unbiased estimator of $F_2(\vec{a}, \vec{p})$, if h is chosen uniformly from a 4-wise independent hash family on $[n]$ with range $\{-1, +1\}$. The estimator can be computed in one-pass and yields a one-pass (ε, δ)-approximation algorithm for computing* REPEAT-RATE *using space and update time of $O(\varepsilon^{-2} \log n \log \delta^{-1})$.*

PROOF. By the previous discussion, $F_2(\vec{b}) = \mathbb{E}_h[\mathrm{Sum2}(h(\vec{b})]$, so the premise of Lemma 11 is satisfied with $Q = F_2$ and $Q' = \mathrm{Sum2}$, so $F_2(\vec{a}, \vec{p}) = \mathbb{E}_h[\mathrm{Sum2}(h(\vec{a}), \vec{p})]$. We now show that $\mathrm{Sum2}(h(\vec{a}), \vec{p})$ can be computed efficiently over the probabilistic stream $(h(\vec{a}), \vec{p})$.

It will be convenient to introduce the jointly independent random variables U_i, for $i \in [m]$ where each U_i equals 1 with probability p_i and equals 0 otherwise. Then

$$\begin{aligned}
\mathrm{Sum2}(h(\vec{a}), \vec{p}) &= \mathbb{E}[(\sum_{i \in [m]} h(a_i) U_i)^2] \\
&= \sum_{i \in [m]} h(a_i)^2 \mathbb{E}[U_i^2] + \sum_{i \neq j} h(a_i) h(a_j) \mathbb{E}[U_i U_j] \\
&= \sum_{i \in [m]} h(a_i)^2 p_i + \sum_{i \neq j} h(a_i) h(a_j) p_i p_j \\
&= (\sum_{i \in [m]} h(a_i) p_i)^2 + \sum_{i \in [m]} p_i (1 - p_i),
\end{aligned}$$

using the fact that $h(a)^2 = 1$. This is easy to compute in one pass.

Finally, we show that this basic estimator can be used to produce a good estimate with high confidence. The approach is a standard one—first we compute the variance of the basic estimator.

$$\mathbb{V}\mathrm{ar}_h[\mathrm{Sum2}(h(\vec{a}), \vec{p})] = \mathbb{V}\mathrm{ar}_h[(\sum_a h(a) P_a)^2], \tag{1}$$

where $P_a = \sum_{i : a_i = a} p_i$ for every $a \in [n]$. This expression is very similar to the one encountered in the variance bound of F_2, except that the values P_a are generalized frequencies that can be non-integral. On the other hand, the same derivation shows that

$$\begin{aligned}
\mathbb{V}\mathrm{ar}_h[(\sum_a h(a) P_a)^2] &= \mathbb{E}_h[(\sum_a h(a) P_a)^4] - (\mathbb{E}_h[(\sum_a h(a) P_a)^2])^2 \\
&= \sum_a P_a^4 + 6 \sum_{a_{i_1} < a_{i_2}} P_{a_{i_1}}^2 P_{a_{i_2}}^2 - (\sum_a P_a^2)^2 \\
&= 4 \sum_{a_{i_1} < a_{i_2}} P_{a_{i_1}}^2 P_{a_{i_2}}^2 \;\leq\; 2(\sum_a P_a^2)^2 \;\leq\; 2\mathbb{E}_h[\mathrm{Sum2}(h(\vec{a}), \vec{p})]^2 \,.
\end{aligned} \tag{2}$$

where the second and last steps use the fact that h is 4-wise independent. Combining Equations 1 and 2 shows that the variance of the estimator is at most twice the square of its expectation. Therefore, the standard technique of taking the average of $O(1/\varepsilon^2)$ such estimators reduces the relative error to ε. As usual the confidence can be increased by taking the median of sufficiently many estimators; we omit the standard calculations. □

5. ESTIMATING MEDIAN

In this section, we present an algorithm for finding an ε-approximate MEDIAN. To solve this problem we use the algorithm of Greenwald and Khanna [Greenwald and Khanna 2001], which approximates the median of a deterministic data stream. Very roughly, their algorithm works as follows: For each item, we calculate the minimum and maximum possible rank within the data seen so far, based on information we have stored; for example, if we have seen $1, 5, 3, 7$, then 6 will have an actual rank of 4, since it is the fourth largest item seen. However, if we only remember that 3 has a minimum rank of 1 and a maximum rank of 3, then all we can say for certain is that the minimum rank of 6 is at least 2 and the maximum rank is at most 5 (since there have been 5 items so far). The value of some of the items, together with their minimum and maximum rank, are stored. Periodically, we sweep through the stored items and remove those that are no longer necessary. So long as the gaps between items' minimum and maximum rank, as well as the gaps between consecutive minimum ranks, is not too large, then the value of any quantile can be approximated well. In particular, we can estimate the value of the median.

Our approach to estimating MEDIAN is based on a deterministic reduction of the problem to median finding in a deterministic stream.

THEOREM 6. *There exists a single pass algorithm finding an ε-approximate* MEDIAN *in $O(\varepsilon^{-1} \log m)$ space.*

PROOF. The idea is use the Greenwald-Khanna selection algorithm [Greenwald and Khanna 2001], on an induced stream A' as follows:

(1) For each tuple $(j, p_i(j))$ in ϑ_i, put $\left\lfloor 2mp_i(j)\varepsilon^{-1} \right\rfloor$ copies of j in A'.
(2) Using the Greenwald-Khanna algorithm, find an element l such that

$$\max\{|\{i \in A' : 1 \le i < l\}|, |\{i \in A' : l < i \le n\}|\} \le (1/2 + \varepsilon/2)|A'| \ . \qquad (3)$$

where $|A'|$ denotes the length of the stream A'.

Note that $p_i(j) \ge \left\lfloor 2mp_i(j)\varepsilon^{-1} \right\rfloor / (2m\varepsilon^{-1}) \ge p_i(j) - \varepsilon/(2m)$. Therefore, dividing Equation 3 by $2m\varepsilon^{-1}$ yields

$$\max\left\{ \left(\sum_{1 \le j < l, i \in [m]} p_i(j) \right), \left(\sum_{l < j \le n, i \in [m]} p_i(j) \right) \right\} - \varepsilon/2 \le (1/2 + \varepsilon/2) \lceil \mathsf{COUNT} \rceil \ .$$

The result follows since $\mathsf{COUNT} > 0$. □

6. ESTIMATING RANGE

In this section, we address the problem of estimating the expected range of a probabilistic stream. We present a deterministic algorithm that uses two passes and

$O(\varepsilon^{-1}\log m)$ space. Our algorithm capitalizes on ideas from [Jayram et al. 2007] in combination with the algorithm for estimating MEDIAN presented in Section 5.

THEOREM 7. *There exists a 2-pass $O(\varepsilon^{-1}\log m)$-space deterministic algorithm to $(1+\varepsilon)$-approximate* RANGE *if* COUNT $> 4C = 4\log(6n/\varepsilon)$.

PROOF. In the first pass we use the MEDIAN estimation algorithm to find a k such that $\mathbb{E}\left[|\{i : X_i \le k\}|\right] \ge C$ and $\mathbb{E}\left[|\{i : X_i \ge k\}|\right] \ge C$. Let A be the event that there exist $i_1, i_2 \in [m]$ such that $X_{i_1} \le k \le X_{i_2}$. Note that i_1 and i_2 need not be distinct if $X_{i_1} = k = X_{i_2}$.

CLAIM 12. $\Pr[A] \ge 1 - \varepsilon/(3n)$.

PROOF. The probability that an element at least as large as k is at least,

$$1 - \prod_{i\in[m]}\left(1 - \sum_{j\ge k} p_i(j)\right) \ge 1 - \exp\left(-\sum_{i\in[m],j\ge k} p_i(j)\right) \ge 1 - \varepsilon/(6n)\ .$$

Similarly, with probability at least $1-\varepsilon/(6n)$, there exists an element that is not-strictly larger than k. The claim follows by the union bound. □

Therefore, with probability at least $1-\varepsilon/(3n)$,

$$\max_i X_i - \min_i X_i + 1 = \max_{i:X_i\ge k}(k - X_i) + \max_{i:X_i\le k}(X_i - k) + 1\ .$$

Hence,

$$\text{RANGE} = (1\pm\varepsilon/3)\left(\mathbb{E}\left[\max_{i:X_i\ge k}(k-X_i)\right] + \mathbb{E}\left[\max_{i:X_i\le k}(X_i-k)\right] + 1\right)\ ,$$

because $1 \le \text{RANGE} \le n$.

Therefore, it suffices to estimate $\mathbb{E}\left[\max_{i:X_i\le k}(k-X_i)\right]$ and $\mathbb{E}\left[\max_{i:X_i\ge k}(X_i-k)\right]$ up to a $(1+\varepsilon/3)$ factor. We can do this in a second pass by utilizing the geometric grouping technique employed in [Jayram et al. 2007]. First consider $\mathbb{E}\left[\max_{i:X_i\le k}(k-X_i)\right]$. Consider the following families of events:

$$\begin{aligned} A_\ell &= \{\forall i \in [m] : X_i > k/(1+\varepsilon/3)^\ell \text{ or } X_i = \perp\} \\ B_\ell &= \{\exists i \in [m] : X_i \in \left(k/(1+\varepsilon/3)^\ell, k/(1+\varepsilon/3)^{\ell-1}\right]\} \end{aligned}$$

and let $q_\ell = \Pr[A_\ell \cap B_\ell] = \Pr[A_\ell]\Pr[B_\ell|A_\ell]$ where $\ell \in [\ell^*]$ and $\ell^* = \left\lceil \log_{1+\varepsilon/3} k \right\rceil$. Note that each q_ℓ can easily be computed in small space as in [Jayram et al. 2007]. Then,

$$\begin{aligned} \mathbb{E}\left[\max_{i:X_i\le k}(k-X_i)\right] &= \sum_{r\le k}\Pr\left[\min_{i:X_i\le k} X_i = r\right](k-r) \\ &= (1\pm\varepsilon/3)\sum_{\ell\in[\ell^*]} q_\ell\left(k - \frac{k}{(1+\varepsilon/3)^\ell}\right)\ . \end{aligned}$$

We estimate $\mathbb{E}\left[\max_{i:X_i\ge k}(X_i-k)\right]$ similarly.

7. COMBINING RELATED OLAP QUERIES

As mentioned in the introduction, OLAP is a multi-dimensional model with dimensions arranged according to some hierarchy. Any single query involves specifying both the values for the dimension attributes, each at some level of the hierarchy, as well as the aggregate operator. In a typical use of an OLAP system, the analyst will not specify one but multiple related OLAP queries. These are typically referred to as *roll-ups* and *drill-downs* since they involve navigating the dimensional hierarchy. For example, using the scenario described in the introduction, the analyst might be interested in computing average sales for every city in California, followed by average sales for California as a whole. Computing these separately will involve multiple scans of the data, and it is desirable to use the results computed in previous queries. For example, can the computation of the queries for each city in California be used to aid the query corresponding to California?

There is a well-known strategy that can be used to speed up the computations for roll-ups and drill-downs on probabilistic data. This is described in detail in [Burdick et al. 2005a] but for the purposes of this paper we can translate this to the setting of probabilistic streams as follows: we are given an aggregator A, and a set of probabilistic data $\{S_1, S_2, \ldots, S_k\}$. The goal is to compute a set of *sufficient statistics* on each S_i so that it is possible to compute A on the composite stream $S_1, S_2, \ldots, S_k$ formed by concatenating the S_i's. For the example given above, every S_i represents the data for a particular city.

It can be seen that the sufficient statistics for computing SUM and COUNT are just their values on the individual probabilistic streams. On the other hand, the solution for MEAN is not that simple. Consider the following approach where the sufficient statistics are just SUM and COUNT. We compute MEAN for the composite stream by taking the ratio of the sum of the SUM values to the sum of the COUNT values. This is valid only when the data is not probabilistic (which is not surprising since otherwise it would yield a simple data stream algorithm for MEAN). We now show how this can be properly achieved for MEAN and other aggregation operators considered in this paper.

For MEAN, observe that it only requires maintaining the values of P_k and A_k for $k = 1, 2, ..., O(\log(1/\varepsilon))$. (For convenience, we assume here that $P \geq 6\ln(2/\varepsilon)$.) To combine one stream with sufficient statistics P_k, A_k with another having P'_k, A'_k, we simply add the corresponding statistics: $P_k + P'_k, A_k + A'_k$. From this, the estimate for MEAN can be generated. Thus the number of sufficient statistics needed for any stream is just $O(\log \varepsilon^{-1})$.

For F_2, the sufficient statistics are just $\sum_i p_i(1 - p_i)$ together with $\sum_i h(a_i)p_i$ for each of the chosen hash functions. (The hash functions chosen must be the same across all streams.) Again, to combine streams, we simply add their corresponding sufficient statistics.

For MIN, the algorithm in [Jayram et al. 2007] divides the domain into a sequence of $O(1/\varepsilon)$ geometrically increasing intervals (bins) and iteratively computes two quantities for each bin: (1) the probability, over possible worlds, that the minimum lies within the bin, and (2) the probability, over possible worlds, that the minimum does not belong to previous bins. The key aspect of that algorithm is that for each item in the input, all that is required of the item is the knowledge of those same

two quantities. Consequently, the sufficient statistics for each stream are just these values computed for all bins. The same principle also applies to MAX; we omit the details.

Although this approach is quite general, it is not obvious how to make this work for combining queries using different aggregators e.g. estimating the MIN of the MEAN of different streams. The key issue seems to be maintaining a succinct data structure that enables fast computation of sufficient statistics corresponding to different aggregators. We leave it as an open problem.

8. CONCLUDING REMARKS

A number of remarkable algorithmic ideas have been developed for estimating aggregates over deterministic streams since the seminal work of [Alon et al. 1999]. Some of them are applicable to estimating aggregates over probabilistic streams such as when estimating MEDIAN and REPEAT-RATE by suitable reductions, but for other aggregates such as DISTINCT and MEAN, we need new ideas that we have presented here.

The probabilistic stream model was initially motivated by probabilistic databases where data items have a distribution associated with them because of the uncertainties and inconsistencies in the data sources. This model has other applications too, including in the motivating scenario we described here in which the stream (topic distribution of search queries) derived from the deterministic input stream (search terms) is probabilistic. We believe that the probabilistic stream model will be very useful in practice in dealing with such applications.

There are several technical and conceptual open problems in addition to the question of combining queries using different aggregators mentioned in the previous section. For example, could one characterize problems for which there is a (deterministic or randomized) reduction from probabilistic streams to deterministic streams without significant loss in space bounds or approximations? We suspect that for additive approximations, there is a simple characterization. Also, can we extend the solutions for estimating the basic aggregates we have presented here to others, in particular, geometric aggregates [Indyk 2004] or aggregate properties of graphs [Feigenbaum et al. 2005a; 2005b]?

9. ACKNOWLEDGEMENTS

We would like to thank Graham Cormode for a helpful comment regarding the results in Section 3. We also thank the anonymous referees for suggestions that improved the presentation of our results.

REFERENCES

Alon, N., Matias, Y., and Szegedy, M. 1999. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences 58,* 1, 137–147.

Babcock, B., Babu, S., Datar, M., Motwani, R., and Widom, J. 2002. Models and issues in data stream systems. *ACM Symposium on Principles of Database Systems*, 1–16.

Balazinska, M., Balakrishnan, H., and Stonebraker, M. 2004. Load management and high availability in the medusa distributed stream processing system. In *ACM International Conference on Management of Data*. 929–930.

Bar-Yossef, Z., Jayram, T., Kumar, R., Sivakumar, D., and Trevisan, L. 2002. Counting distinct elements in a data stream. In *Proc. 6th International Workshop on Randomization and Approximation Techniques in Computer Science*. 1–10.

Batu, T., Kannan, S., Khanna, S., and McGregor, A. 2004. Reconstructing strings from random traces. In *ACM-SIAM Symposium on Discrete Algorithms*. 910–918.

Burdick, D., Deshpande, P., Jayram, T. S., Ramakrishnan, R., and Vaithyanathan, S. 2005. OLAP over uncertain and imprecise data. In *International Conference on Very Large Data Bases*. 970–981.

Burdick, D., Deshpande, P. M., Jayram, T. S., Ramakrishnan, R., and Vaithyanathan, S. 2006. Efficient allocation algorithms for OLAP over imprecise data. In *International Conference on Very Large Data Bases*. 391–402.

Chang, K. L. and Kannan, R. 2006. The space complexity of pass-efficient algorithms for clustering. In *ACM-SIAM Symposium on Discrete Algorithms*. 1157–1166.

Cormode, G. and Garofalakis, M. 2007. Sketching probabilistic data streams. In *ACM International Conference on Management of Data*.

Cranor, C. D., Johnson, T., Spatscheck, O., and Shkapenyuk, V. 2003. Gigascope: A stream database for network applications. In *ACM International Conference on Management of Data*. 647–651.

Dalvi, N. N. and Suciu, D. 2004. Efficient query evaluation on probabilistic databases. In *International Conference on Very Large Data Bases*. 864–875.

Feigenbaum, J., Kannan, S., McGregor, A., Suri, S., and Zhang, J. 2005a. Graph distances in the streaming model: the value of space. In *ACM-SIAM Symposium on Discrete Algorithms*. 745–754.

Feigenbaum, J., Kannan, S., McGregor, A., Suri, S., and Zhang, J. 2005b. On graph problems in a semi-streaming model. *Theoretical Computer Science 348,* 2-3, 207–216.

Flajolet, P. and Martin, G. N. 1985. Probabilistic counting algorithms for data base applications. *J. Comput. Syst. Sci. 31,* 2, 182–209.

Fuhr, N. and Roelleke, T. 1997. A probabilistic relational algebra for the integration of information retrieval and database systems. *ACM Trans. Inf. Syst. 15,* 1, 32–66.

Greenwald, M. and Khanna, S. 2001. Efficient online computation of quantile summaries. In *ACM International Conference on Management of Data*. 58–66.

Guha, S. and McGregor, A. 2007. Space-efficient sampling. In *AISTATS*. 169–176.

Indyk, P. 2004. Algorithms for dynamic geometric problems over data streams. In *ACM Symposium on Theory of Computing*. 373–380.

Jayram, T. S., Kale, S., and Vee, E. 2007. Efficient aggregation algorithms for probabilistic data. In *ACM-SIAM Symposium on Discrete Algorithms*.

Kannan, S. and McGregor, A. 2005. More on reconstructing strings from random traces: Insertions and deletions. In *IEEE International Symposium on Information Theory*. 297–301.

Lipson, J. D. 1981 *Elements of algebra and algebraic computing*. Addison-Wesley Publishing Company.

Munro, J. I. and Paterson, M. 1980. Selection and sorting with limited storage. *Theor. Comput. Sci. 12*, 315–323.

Muthukrishnan, S. 2006. *Data Streams: Algorithms and Applications*. Now Publishers.

Appendix

Claim 9. *For all $k > 0$ and for $\varepsilon < 1/2$,*

$$\prod_{i=1}^{k}(1+\frac{1}{i2^i}\varepsilon) \leq (1+\frac{k}{k+1}\varepsilon) \ .$$

PROOF. We proceed by induction. The base case $k = 1$ follows immediately. Furthermore, for $k = 2$ we have $(1+\varepsilon/2)\,(1+\varepsilon/8) = 1+5\varepsilon/8+\varepsilon^2/16 \leq 1+2\varepsilon/3$, where the last inequality follows because $\varepsilon \leq 1/2$. So assume the statement is true for some $k-1 \geq 2$, and consider the case for k. With a little foresight, we first note that for $k \geq 2$,

$$(k+1)(3k-1) \leq k2^{k+1} \ . \tag{4}$$

Thus,

$$\begin{aligned}(k+1)[(k-1)k2^{k+1}+2k+(k-1)] &= (k^3-k)2^{k+1}+(k+1)(3k-1) \ \text{ by (4)}\\ &\leq (k^3-k)2^{k+1}+k2^{k+1}\\ &= k^3 2^{k+1} \ .\end{aligned}$$

Dividing both sides by $(k+1)k^2 2^{k+1}$, we find

$$\frac{k}{k-1}+\frac{1}{k2^k}+\frac{1}{2}\frac{k-1}{k}\frac{1}{k2^k} \ \leq \ \frac{k}{k+1} \ . \tag{5}$$

Thus, we have

$$\begin{aligned}\prod_{i=1}^{k}(1+\frac{1}{i2^i}\varepsilon) &= (1+\frac{\varepsilon}{k2^k})\prod_{i=1}^{k-1}(1+\frac{1}{i2^i}\varepsilon)\\ &\leq \left(1+\frac{\varepsilon}{k2^k}\right)\left(1+\frac{k-1}{k}\varepsilon\right) \ \text{ by induction.}\\ &= 1+\varepsilon\left(\frac{k-1}{k}+\frac{1}{k2^k}+\varepsilon\frac{k-1}{k}\frac{1}{k2^k}\right)\\ &\leq 1+\varepsilon\left(\frac{k-1}{k}+\frac{1}{k2^k}+\frac{1}{2}\frac{k-1}{k}\frac{1}{k2^k}\right)\\ &\leq 1+\frac{k}{k+1}\varepsilon \ \text{ by (5) .}\end{aligned}$$

□

Received October 2007; accepted ?????