Moment-Based Quantile Sketches for Efficient High Cardinality Aggregation Queries

Edward Gan, Jialin Ding, Kai Sheng Tai, Vatsal Sharan, Peter Bailis
Stanford InfoLab

ABSTRACT

Interactive analytics increasingly involves querying for quantiles over specific sub-populations and time windows of high cardinality datasets. Data processing engines such as Druid and Spark use mergeable summaries to estimate quantiles on these large datasets, but summary merge times are a bottleneck during high-cardinality aggregation. We show how a compact and efficiently mergeable quantile sketch can support aggregation workloads. This data structure, which we refer to as the moments sketch, operates with a small memory footprint (200 bytes) and computationally efficient (50ns) merges by tracking only a set of summary statistics, notably the sample moments. We demonstrate how we can efficiently and practically estimate quantiles using the method of moments and the maximum entropy principle, and show how the use of a cascade further improves query time for threshold predicates. Empirical evaluation on real-world datasets shows that the moments sketch can achieve less than 1 percent error with 40× less merge overhead than comparable summaries, improving end query time in the MacroBase engine by up to 7× and the Druid engine by up to 60×.

1 INTRODUCTION

Performing interactive multi-dimensional analytics over data from sensors, devices, phones, and servers increasingly requires computing aggregate statistics for specific subpopulations and time windows [4, 29, 62]. In applications such as A/B testing [36, 40], exploratory data analysis [9, 70], and operations monitoring [2, 15], analysts frequently perform cube-like aggregate queries to better understand how particular user cohorts, device types, and feature flags are behaving. In particular, computing quantiles over these subpopulations is an essential part of debugging and real-time monitoring workflows [26].

As an example of this quantile-driven analysis, our industry collaborators collect billions of telemetry events daily from millions of heterogeneous mobile devices. Each device tracks multiple metrics including request latency and memory usage, and each device is associated with dimensional metadata such as application version and hardware model. The company's engineers and analysts issue quantile queries over this telemetry by aggregating by different dimensions to both monitor their application (e.g., examine memory trends across time windows) and debug regressions (e.g., examine tail latencies across versions).

Performing full scans over raw metrics can be expensive in terms of both network bandwidth and storage, so in-memory OLAP engines such as Druid [64, 77] and Spark [37, 78] instead compute approximate quantiles from a compressed representation of the data values. By pre-aggregating a summary for each combination of dimension values, an engine like Druid can reduce query times by aggregating the relevant summaries directly, effectively operating over a summary data cube [32, 62]. Figure 1 illustrates how, if such

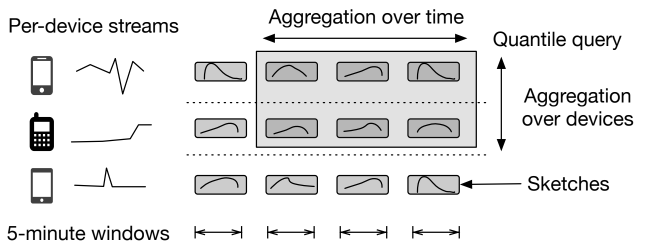

Figure 1: Queries across multiple devices and time ranges can be performed by merging the relevant pre-aggregated summaries.

summaries are *mergeable* [3], they can be aggregated without loss of accuracy. Engines can subsequently calculate quantiles across specific populations and time ranges by merging the summaries directly.

A variety of existing mergeable quantile summaries [3, 28, 33] enable this functionality, but their merge overheads can lead to severe performance penalties on high-cardinality datasets. In our example telemetry deployment, which maintains hundreds of thousands of dimension combinations at a five-minute granularity, calculating a quantile over a given 2-week range can require merging millions of summaries. Based on our experiments (Section 6.2.1), one million 1KB GK-sketches [33] require more than 10 seconds to merge, limiting the types of queries users can ask interactively. For long-standing and recurring queries, materialized views [42, 47, 58] and sliding window sketches [25] can reduce this overhead. However, the combinatorial explosion of dimensions and the cost of maintaining multiple views means merge time remains an important bottleneck. Our industry collaborators confirm this need, and report merges of ten million sketches in production.

In this paper, we enable interactive quantile queries over high-cardinality aggregates by introducing a compact and efficiently mergeable quantile sketch and associated quantile estimation routines. We draw a connection between the classic *method of moments* for parameter estimation in statistics [74] and the need for efficient summary data structures. We show that storing the sample moments $\mu_i = \frac{1}{n}\sum x^i$ and log-moments $\nu_i = \frac{1}{n}\sum \log^i(x)$ can enable accurate quantile estimation over a range of real-world datasets while utilizing fewer than 200 bytes of memory and incurring merge times of less than 50 nanoseconds. In the context of mergeable quantile estimation, we refer to our proposed summary data structure as the *moments sketch*.

While constructing the moments sketch is straightforward, the inverse problem of estimating quantiles from the summary is more complex. The statistics in a moments sketch provide only loose constraints on the distribution of values in the original dataset: many distributions might match the moments of a moments sketch but fail to capture the dataset structure. Therefore, we make use of the *principle of maximum entropy* [39] to compute a "least-biased" quantile estimate for a moments sketch. Empirically, we find that this approach yields more accurate estimates than alternative methods, achieving $\epsilon \leq 1\%$ error with 200 bytes of memory. To achieve this, we also describe a series of practical optimizations to standard entropy maximization that allow us to compute quantile estimates in under 1 millisecond on a range of real-world datasets.

Moving beyond simple quantile queries, many complex queries depend on the quantile estimates of multiple subpopulations. For example, data exploration systems such as MacroBase [9] are interested in finding all subpopulations that match a given threshold condition (e.g., subpopulations where the 95th percentile latency is greater than the global 99th percentile latency). Given a large number of subpopulations, the cost of millisecond-level quantile estimates on thousands of subgroups will accumulate. Therefore, to support threshold queries over multiple populations, we extend our quantile estimator with a *cascade* [71], or sequence of increasingly precise and increasingly expensive estimates based on bounds such as the Markov inequalities. For queries with threshold conditions, the cascades dramatically reduce the overhead of quantile estimation in a moments sketch, by up to 25×.

We implement the moments sketch both as a reusable library and as part of the Druid and MacroBase analytics engines. We empirically compare its accuracy and efficiency with alternative mergeable quantile summaries on a variety of real-world datasets. We find that the moments sketch offers 40 to 200× faster merge times than alternative summaries with comparable accuracy. This enables 35 to 150× faster query times on real datasets. Moreover, the moments sketch enables up to 7× faster analytics queries when integrated with MacroBase and 60× faster end-to-end queries when integrated with Druid.

In summary, we make the following contributions:

- We illustrate how statistical moments are useful as efficient mergeable quantile sketches in aggregation and threshold-based queries over high-cardinality data.
- We demonstrate how statistical and numerical techniques allow us to solve for accurate quantile estimates in less than 1 ms, and show how the use of a cascade further improves estimation time on threshold queries by up to 25×.
- We evaluate the use of moments as quantile summaries on a variety of real-world datasets and show that the moments sketch enables 35 to 150× faster query times in isolation, up to 7× faster queries when integrated with MacroBase and up to 60× faster queries when integrated with Druid over comparably-accurate quantile summaries.

The remainder of this paper proceeds as follows. In Section 2, we discuss related work. In Section 3, we review relevant background material. In Section 4, we describe the proposed moments sketch. In Section 5, we describe a cascade-based approach for efficiently answering threshold-based queries. In Section 6, we evaluate the moments sketch in a series of microbenchmarks. In Section 7, we evaluate the moments sketch as part of the Druid and MacroBase systems, and measure its performance in sliding window and low-precision environments. We conclude in Section 8.

2 RELATED WORK

High-performance aggregation. The aggregation scenarios in Section 1 are found in many existing streaming data systems [9, 17, 24, 62, 77], as well as data cube [32, 65], data exploration [2], and visualization [18] systems. In particular, these systems can perform interactive aggregations over time windows and along many cube dimensions, motivating the design of our sketch. Many of these systems use approximate query processing [4, 34, 55] and use sampling and summaries to improve query performance, but do not address data structures specific to quantiles. We believe the moments sketch serves as a useful primitive in these engines.

Sensor networking is a rich source of algorithms for heavily resource-constrained settings. Sensor network aggregation systems [49] support large scale roll-ups, but seminal work in this area is largely focused on the complementary problem of communication plans over a network [21, 43, 50]. Mean, min, max, and standard deviation in particular are used in [49] as functions amenable to computation-constrained environments, but the authors do not consider higher moments or their application to quantile estimation.

Several database systems make use of summary statistics in general-purpose analytics. Muthukrishan et al [57] observe that the moments are a convenient statistic in streaming settings and use it to fill in missing integers. Data Canopy [73] uses first and second moments as an efficiently mergeable statistic for computing standard deviations and linear regressions. Similarly, systems on probabilistic data cubes such as [76] use the first and second moments to prune queries over cube cells that store distributions of values. In addition, many methods use compressed data representations to perform statistical analyses such as linear regression, logistic regression, and PCA [20, 60, 66, 75]. We are not aware of prior work utilizing higher moments to efficiently estimate quantiles for high-dimensional aggregation queries.

Quantile summaries. There are a variety of summary data structures for the ϵ-approximate quantile estimation problem [19, 23, 33, 48, 67]. Some of these summaries assume values from a finite universe [23, 67], while others operate using only comparisons [3, 33]. Our proposed moments sketch and others [13, 28] operate on real values. Agarwal et al. [3] provide the initial motivation for mergeable summaries, as well as a proposed mergeable quantile sketch. To our knowledge we are the first to evaluate quantile sketches for the purposes of sub-microsecond merge overheads.

Method of moments. The method of moments is a well-established statistical technique for estimating the parameters of probability distributions [74]. The main idea behind this approach is that the parameters of a distribution of interest P can be related to the expectations of functions of the random variable $X \sim P$. As a general method for consistent statistical parameter estimation, the method of moments is used across a wide range of fields, including econometrics [35], physics [31, 53], and machine learning [7, 12, 41]. In this work, we demonstrate how the method of moments, applied in conjunction with practical performance optimizations, can scale to support real-world latency-sensitive query processing workloads.

Maximum entropy principle. The maximum entropy principle prescribes that one should select the least informative distribution that is consistent with the observed data. In the database community, this principle has been applied to estimating cardinality [69] and predicate selectivity [51]. Mead and Papanicolaou [53] apply the maximum entropy principle to the problem of estimating distributions subject to moment constraints; follow-up work proposes the use of Chebyshev polynomials for stability [11, 68] and faster approximation algorithms [10], though we have not seen any practical implementations suitable for use in a database. The maximum entropy principle is also used in machine learning, notably in the context of *maximum entropy models* [14]. For example, in natural language processing, maximum entropy models are a popular choice for tasks such as text classification [59] and sequence tagging [45].

3 BACKGROUND

In this section, we review approximate quantile estimates, mergeable quantile summaries, and our target query model.

3.1 Quantile Queries

Given a dataset D with n elements, for any $\phi \in (0, 1)$, the *ϕ-quantile* is the item $x \in D$ with rank $r(x) = \lfloor \phi n \rfloor$, where the rank of x is the number of elements in D smaller than x. Computing exact quantiles in a single pass requires memory linear in the size of the dataset [56]. This limits the practicality of computing exact quantiles on large datasets, so we instead focus on approximate quantiles.

An ϵ-approximate ϕ-quantile is an element with rank between $(\phi - \epsilon)n$ and $(\phi + \epsilon)n$ [3]. Given an estimated ϕ-quantile q_ϕ, we can also define its *quantile error* ε [48] as the following:

$$\varepsilon = \frac{1}{n} \left| \text{rank}\left(q_\phi\right) - \lfloor \phi n \rfloor \right|, \qquad (1)$$

such that an ϵ-approximate quantile has error at most ε. For example, given a dataset $D = \{1, \ldots, 1000\}$, a quantile estimate $q_{0.5} = 504$ for $\phi = 0.5$ would have error $\varepsilon = 0.003$. In this paper, we consider datasets D represented by collections of by real numbers $x \in \mathbb{R}$.

Quantile summaries are data structures that provide approximate quantile estimates for a dataset given space sub-linear in n. These summaries usually have a parameter k_ϵ that trades off between the size of the summary and the accuracy of its estimates. An ϵ-approximate quantile summary provides ϵ approximate ϕ-quantiles, where ϵ can be a function of space usage and the dataset [19, 23, 33, 67].

3.2 Mergeable Summaries

Agarwal et al. [3] introduce the concept of *mergeability* to accurately combine summaries in distributed settings. Formally, for a summary with parameter k_ϵ, we use $S(D, k_\epsilon)$ to denote a valid summary for a dataset D. For any pair of datasets D_1 and D_2, the summarization routine S is *mergeable* if there exists an algorithm (i.e., the "merge" procedure) that produces a combined summary

$$S(D_1 \uplus D_2, k_\epsilon) = \text{merge}(S(D_1, k_\epsilon), S(D_2, k_\epsilon))$$

from any two input summaries, where $\uplus$ denotes multiset addition.

Intuitively, a summary is mergeable if there is no accuracy cost to incrementally computing it over subsets of data. Thus, mergeable summaries are *algebraic* aggregate functions in the data cube literature [32]. As an example, an equi-depth histogram [22] on its own is not mergeable because there is no way to accurately combine two overlapping histogram buckets without access to additional data.

Mergeable summaries are motivated by distributed systems such as MapReduce: a "map" function can be naturally used to construct summaries while a "reduce" function merges sketches to calculate an accurate summary over a large dataset [3]. While a MapReduce job may require a merge for every mapper, a query over a data cube can require a merge for each of the potentially millions of cells in the cube. Any accuracy losses incurred by merges would compound, so mergeability is essential in this context.

3.3 Query Model

As described in Section 1, we focus on improving the performance of quantile queries over aggregations on high cardinality datasets. Given a dataset with d categorical dimensions, we consider data cubes that maintain summaries for every d-way dimension value tuple as one natural setting for high performance aggregations, and many other settings are also applicable [73]. In these settings, query time is heavily dependent on the number of merges and the time per merge.

We consider two broad classes of queries in this paper. First, *single quantile* queries ask for quantile estimates for a single specified population. For example, we can query the 99th percentile of latency over the last two weeks for a given version of the application:

```
SELECT percentile(latency, 99) FROM requests
WHERE time > date_sub(curdate(), 2 WEEK)
AND app_version = "v8.2"
```

To process this query in time t_{query}, we would need to merge n_{merge} summaries, each with runtime overhead t_{merge}, and then estimate the quantile from the merged summary with runtime cost t_{est}. This results in total query time:

$$t_{\text{query}} = t_{\text{merge}} \cdot n_{\text{merge}} + t_{\text{est}}. \qquad (2)$$

We evaluate the different regimes where queries are bottlenecked on merges and estimation in Figure 6 in Section 6.2.1: merge time begins to dominate at around $n_{\text{merge}} \geq 10^4$. We consider loading the summaries and network communication as additional potential components to the merge time.

We also consider *threshold* queries which are conditioned on sub-groups or windows with percentiles above a specified threshold. For example, we may be interested in combinations of application version and hardware platform for which the 99th percentile latency exceeds 100ms:

```
SELECT app_version, hw_model,
    PERCENTILE(latency, 99) as p99
FROM requests WHERE p99 > 100
GROUP BY app_version, hw_model
```

These queries have additional runtime cost that depends on the number of groups n_{groups} we operate over since t_{est} can be significant when one is searching for high quantiles over thousands of sub-groups. This results in total query time:

$$t_{\text{query}} = t_{\text{merge}} \cdot n_{\text{merge}} + t_{\text{est}} \cdot n_{\text{groups}}. \qquad (3)$$

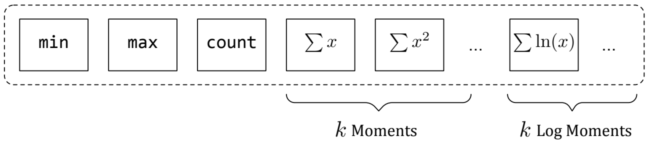

Figure 2: The moments sketch is an array of floating point values.

Algorithm 1: Moments sketch construction

```
input: number of moments $k$
initialization
    $x_{\min}, x_{\max} \leftarrow \infty, -\infty$
    $\vec{\mu}, \vec{\nu} \leftarrow \vec{0}, \vec{0}$
    $n \leftarrow 0$
function Update($x$)
    $x_{\min} \leftarrow \min\{x, x_{\min}\}$
    $x_{\max} \leftarrow \max\{x, x_{\max}\}$
    $n \leftarrow n + 1$
    for $i \in \{1, \ldots, k\}$ do
        $\mu_i \leftarrow \frac{n-1}{n}\mu_i + \frac{1}{n}x^i$                ▷ Standard moments
        if $x > 0$ then
            $\nu_i \leftarrow \frac{n-1}{n}\nu_i + \frac{1}{n}\log^i(x)$      ▷ Log-moments
```

As described in Section 1, such queries are very useful for debugging and data exploration.

4 THE MOMENTS SKETCH

In this section, we describe how we perform quantile estimation using the moments sketch. First, we review the summary statistics stored in the moments sketch and describe how they comprise an efficiently mergeable sketch. Second, we describe how we can use the method of moments and the maximum entropy principle to estimate quantiles from the moments sketch, with details on how we resolve practical difficulties with numerical stability and estimation time. We conclude with a discussion of theoretical guarantees on the approximation error of quantiles estimated from the sketch.

4.1 Moments Sketch Statistics

The moments sketch of a dataset D is a set of floating point values: the minimum value $x_{\min}$, the maximum value $x_{\max}$, the count n, the sample moments $\mu_i = \frac{1}{n}\sum_{x \in D} x^i$ and the sample logarithmic moments $\nu_i = \frac{1}{n}\sum_{x \in D} \log^i(x)$ for $i \in \{1, \ldots, k\}$ (Figure 2). The moments sketch has an integer parameter k, which describes the highest power used in the moments. We refer to k as the *order* of a moments sketch. Each sample moment provides additional information about the distribution, so higher-order moments sketches are more precise but have higher space and computation time overheads.

We construct a moments sketch for a dataset D using either streaming point-wise updates (Algorithm 1) or by merging existing moments sketches. For each data point, we update the minimum and maximum, then accumulate the counts and moments. As an implementation detail, we can accumulate the unscaled sums $\sum x^i$ and $\sum \log^i(x)$ instead of the μ_i, ν_i directly. We merge two moments sketches with the same k by combining the minimum, maximum, count, and the moments via comparison and potentially vectorized addition.

Log-moments. The moments sketch records logarithmic moments (log-moments) in order to recover better quantile estimates for long-tailed datasets. In particular, taking the logarithm of data points is useful when values in the dataset can vary over several orders of magnitude. In general, when updating a moments sketch in a streaming manner or when maintaining multiple moments sketches in a distributed setting, we cannot know *a priori* whether standard moments or log-moments are more appropriate for the given dataset. Therefore, our default approach is to store both sets of moments up to the same order k. Given additional prior knowledge of the data distribution, we may choose to maintain a moments sketch with only a single set of moments.

Data points with negative values pose a potential problem for the log-moments since the logarithm is undefined for these points. There are several strategies for addressing this, including storing separate sums for positive and negative values and shifting all values so that they are positive. In this paper, we adopt the simple approach of ignoring the log sums when there are any negative values, and computing estimates using the remaining statistics.

Remark on pathological distributions. We distinguish between the sample moments tracked by the moments sketch, and the moments of a distribution with density $f(x)$ given by $\int x^i f(x)dx$. The moments of certain "pathological" distributions may be undefined; for example, the standard Cauchy distribution $f(x) = \pi^{-1}\left(1 + x^2\right)^{-1}$ does not have finite moments of any order. However, the moments sketch tracks the moments of the empirical distribution, which are always well-defined. Note that this suits our goal of estimating quantiles on the empirical distribution.

4.2 Estimating Quantiles

Method of moments. To estimate quantiles from a moments sketch, we apply the *method of moments* [7, 12, 41, 74] to construct a distribution $f(x)$ whose moments match those recorded in the sketch. Specifically, given a moments sketch with minimum $x_{\min}$ and maximum $x_{\max}$, we solve for a pdf $f(x)$ supported on $[x_{\min}, x_{\max}]$ such that $\int_{x_{\min}}^{x_{\max}} x^i f(x)\,dx = \mu_i$ and $\int_{x_{\min}}^{x_{\max}} \log^i(x) f(x)\,dx = \nu_i$. We can then report the quantiles of $f(x)$ as estimates for the quantiles of the dataset.

In general, a finite set of moments does not uniquely determine a distribution [5]. That is, there are often many possible distributions with varying quantiles that each match a given set of sample moments. Therefore, we must disambiguate between them.

Maximum entropy. In this work, we use the *principle of maximum entropy* [39] to select a unique distribution that satisfies the given moment constraints. Intuitively, the differential Shannon entropy H of a distribution with pdf $f(x)$, defined as $H[f] = -\int_{\mathcal{X}} f(x)\log f(x)\,dx$, is a measure of the degree of *uninformativeness* of the distribution. For example, a uniform distribution has higher entropy than a point mass distribution. Thus, the maximum entropy distribution can be seen as the distribution that encodes the *least* additional information about the data beyond that captured by the given moment constraints.

Applying the maximum entropy principle to the moments sketch, we estimate quantiles by solving for the pdf f that maximizes the entropy while matching the moments in the sketch. Following, we estimate quantiles using numeric integration and the Brent's method for root finding [61] .

In practice, we find that the use of maximum entropy distributions yields quantile estimates with comparable accuracy to baseline methods on a range of real-world datasets. We discuss our empirical results further in Section 6.2.3.

Optimization. We now describe how to solve for the maximum entropy distribution f. We trade off between accuracy and estimation time by solving for f subject to the first k_1 standard moments and k_2 log-moments stored in the sketch; incorporating more moments leads to more precise estimates but more computationally expensive estimation. As previously noted, for datasets with non-positive values (i.e., $x_{\min} \le 0$), we set $k_2 = 0$. Therefore, our goal is to find the solution f to the following optimization problem:

$$\begin{aligned} \underset{f \in \mathcal{F}[x_{\min}, x_{\max}]}{\text{maximize}} \quad & H[f] && (4) \\ \text{subject to} \quad & \int_{x_{\min}}^{x_{\max}} x^i f(x)\,dx = \mu_i, && i \in \{1, \dots, k_1\} \\ & \int_{x_{\min}}^{x_{\max}} \log^i(x) f(x)\,dx = \nu_i, && i \in \{1, \dots, k_2\} \end{aligned}$$

where $\mathcal{F}[x_{\min}, x_{\max}]$ denotes the set of distributions supported on $[x_{\min}, x_{\max}]$.

It is well known that the solution to Problem 4 is a member of the class of exponential family distributions [39]:

$$f(x;\theta) = \exp\left(\theta_0 + \sum_{i=1}^{k_1} \theta_i x^i + \sum_{i=1}^{k_2} \theta_{k_1+i} \log^i(x)\right),$$

where θ_0 is a normalization constant such that $f(x;\theta)$ integrates to 1 over the domain $[x_{\min}, x_{\max}]$. Finding the distribution $f(x;\theta)$ that satisfies the moment constraints is a convex optimization problem (over the parameters θ) that we solve using Newton's method [53].

4.3 Practical Implementation

In this section, we outline implementation details that are important for solving the constrained entropy maximization problem in practice. Let $L(\theta)$ denote the objective that we optimize using Newton's method. The Hessian of L takes the form:

$$\nabla^2 L(\theta)_{ij} = \int_{x_{\min}}^{x_{\max}} m_i(x) m_j(x) f(x;\theta)\,dx, \qquad (5)$$

where the functions $m_i(x)$ range over the set of functions

$$\{x^i : i \in \{1, \dots, k_1\}\} \cup \{\log^i(x) : i \in \{1, \dots, k_2\}\}.$$

There are two main challenges in computations involving the Hessian in each Newton step. First, $\nabla^2 L$ can be nearly singular and cause numerical instabilities in Newton's method that prevent or slow down convergence. Second, since the integral in Eq. 5 has no closed form, the cost of performing $O(k^2)$ numerical integrations to compute $\nabla^2 L$ in each iteration can be expensive. We describe our solutions to each of these issues in turn, and then discuss how they influence our choice of k and k_1, k_2 during optimization.

Numerical Stability. To quantify the degree of numerical instability, we use the *condition number* of the matrix $\nabla^2 L$. The condition number $\kappa(A)$ of a matrix A describes how close a matrix is to being singular: matrices with high condition number are close to being singular, and $\log_{10} \kappa$ provides an estimate of how many digits of precision are lost when inverting A. In particular, the use of the powers $m_i(x) \in \{x^i : i \in \{1, \dots, k_1\}\}$ can result in ill-conditioned Hessians [30]. For example, when solving for a maximum entropy distribution with $k_1 = 8, k_2 = 0, x_{\min} = 20$, and $x_{\max} = 100$, we encountered $\kappa \approx 3 \cdot 10^{31}$ at $\theta = 0$, making even the very first Newton step infeasible using double precision.

We mitigate this issue by using a change of basis from the functions $m_i(x) = x^j$ and $m_i(x) = \log^j(x)$ to the basis of Chebyshev polynomials $T_i(x)$ [11, 61]. We define the new basis $\tilde{m}_i$ as follows:

$$\tilde{m}_i(x) = \begin{cases} T_i(s_1(x)), & i \in \{1, \dots, k_1\} \\ T_{i-k_1}(s_2(\log(x))), & i \in \{k_1 + 1, \dots, k_1 + k_2\} \end{cases}$$

where s_1, s_2 are linear scaling functions to map the domain to $[-1, 1]$. The new basis functions $\tilde{m}_i(x)$ can be expressed in terms of x^j and $\log^j(x)$ using standard formulae for Chebyshev polynomials and the binomial expansion [52]. Using this new basis for the same problem instance as above, we found that the condition number was reduced to $\kappa \approx 11.3$, making precision loss during each Newton step less of a concern.

Efficient Integration. Naïvely computing the Hessian requires evaluating $O(k^2)$ numerical integrals per iteration, which can lead to prohibitively slow estimation time. We reduce this computational overhead by using polynomial approximations of the functions appearing in the integrands. The use of these of these polynomial approximations allows us to compute the integrals efficiently by obviating the need for expensive numerical integration.

Observe that if the integrands $\tilde{m}_i(x)\tilde{m}_j(x)f(x;\theta)$ were expressible as polynomials in x, then each integral can be evaluated in closed form. The factors in the integrand that do not appear as polynomials in x are $\tilde{m}_i(x), i \in \{k_1 + 1, \dots, k_1 + k_2\}$, which are polynomials in $\log(x)$, and the pdf $f(x;\theta)$. Therefore, we compute Chebyshev polynomial approximations of these factors using a fast cosine transform [61] and replace each instance in the integrands with its corresponding approximation.[1] Using this method, each Hessian evaluation thus requires $k_2 + 1$ polynomial approximations.

In practice, this approximation is crucial to achieving fast estimation time. As we show in our empirical evaluation (Section 6.3), this polynomial approximation strategy reduces estimation time by over 20× compared to the alternative of running numerical integration to compute each entry. We find in our experiments that the major bottleneck during maximum entropy optimization is constructing the $k_2 + 1$ polynomial approximations and then symbolically computing the $O((k_1 + k_2)^2)$ entries in the Hessian.

Choosing k. The two optimizations we have described improve the stability and runtime of our maximum entropy solver. However, for large k these can still be an issue. In this section we discuss how k affects the operation of our estimator implementation.

[1]This approach is similar to the idea behind weighted Clenshaw-Curtis integration [61].

Given a moments sketch with k moments and log moments, users can configure how many of each k_1, k_2 they would like to make use of when estimating quantiles. In our implementation we use a set of default heuristics to select k_1, k_2 with condition number below a threshold $\kappa_{\max}$. Our heuristics greedily increment k_1 and k_2, favoring moments where we do not detect precision loss and which are closer to the moments expected from a uniform distribution.

The precision of the higher moments is also limited for large k. The moments sketch does not track Chebyshev polynomial moments directly since Chebyshev polynomials are difficult to rescale when merging summaries with different $x_{\min}, x_{\max}$. Instead, we calculate them from the moments μ_i and ν_i, incurring numeric precision loss in the process since the conversion relies on cancellation between the μ_i, ν_i. The precision loss is significant on some datasets when $k \geq 15$ (e.g. Fig. 9 in Sec. 6.2.3), so in this paper we focus on the regime where $k < 15$.

If the dataset range is fixed then storing Chebyshev polynomial moments directly in a moments sketch is an alternative. Furthermore, if users have the space budget, the moments sketch can be extended with moments of additional transformations such as the power transforms $x' = x^{1/p}$. We view the development of hybrid sketches that supplement the moments sketch with additional statistics and data structures as an interesting area for future work.

4.4 Quantile Error Bounds

Recall that we estimate quantiles using the maximum entropy distribution subject to the moment constraints recorded in the moments sketch. Since the true empirical distribution is in general not equal to the estimated maximum entropy distribution, to what extent can the quantiles estimated from the sketch deviate from the true quantiles? In this section, we discuss worst-case bounds on the discrepancy between *any* two distributions which share the same moments, and relate these to bounds on the quantile estimate errors. In practice, error on non-adversarial datasets is lower than these bounds suggest.

We consider distributions supported on $[-1, 1]$: we can scale and shift any distribution with bounded support to match. By Proposition 1 in Kong and Valiant [44], any two distributions supported on $[-1, 1]$ with densities f and g and standard moments μ_f, μ_g, the Wasserstein distance (or Earth Mover's distance) $W_1(f, g)$ between f and g is bounded by:

$$W_1(f,g) \leq O\left(\frac{1}{k} + 3^k \|\mu_f - \mu_g\|_2\right).$$

For univariate distributions f and g, the Wasserstein distance between the distributions is equal to the L1 distance between their respective cumulative distribution functions F and G (see Theorem 6.0.2 in [6]). Thus:

$$W_1(f,g) = \int_{-1}^{+1} |F(x) - G(x)|\, dx.$$

If f is the true dataset distribution, we estimate q_ϕ by calculating the ϕ-quantile of the maximum entropy distribution $\hat{f}$. The quantile error $\varepsilon(q_\phi)$ is then equal to the gap between the CDFs: $\varepsilon(q_\phi) = |F(q_\phi) - \hat{F}(q_\phi)|$. Therefore, the average quantile error over the support $[-1, 1]$ is bounded as follows:

$$\int_{-1}^{+1} \varepsilon(x)\, dx \leq O\left(\frac{1}{k} + 3^k \|\mu_f - \mu_{\hat{f}}\|_2\right). \qquad (6)$$

Since we can run Newton's method until the moments μ_f and $\mu_{\hat{f}}$ match to any desired precision, the $3^k \|\mu_f - \mu_{\hat{f}}\|_2$ term is negligible.

Equation 6 does not directly apply to the ϵ_{avg} used in Section 6, which is averaged over ϕ for uniformly spaced ϕ-quantiles rather than over the support of the distribution. Since $q_\phi = \hat{F}^{-1}(\phi)$ and $\phi = \hat{F}(x)$, we can relate ϵ_{avg} to Eqn 6 using our maximum entropy distribution $\hat{f}$:

$$\int_0^1 \varepsilon(q_\phi)\, d\phi = \int_{-1}^{+1} \varepsilon(x)\hat{f}(x)\, dx \leq O\left(\frac{\hat{f}_{\max}}{k}\right)$$

where $\hat{f}_{\max}$ is the maximum density of our estimate. Thus, we expect the average quantile error ϵ_{avg} to have a decreasing upper bound as k increases, with higher potential error when $\hat{f}$ has regions of high density relative to its support. Though these bounds are too conservative to be useful in practice, they provide useful intuition on how worst case error can vary with k and $\hat{f}$ (c.f. Figure 8).

5 THRESHOLD QUERIES

We described in Section 3.3 two types of queries: single quantile queries and threshold queries over multiple groups. The optimizations in Section 4.3 can bring quantile estimation overhead down to $\leq$ 1ms, which is sufficient for interactive latencies on single quantile queries. In this section we show how we can further reduce quantile estimation overheads on threshold queries. Instead of computing the quantile on each sub-group directly, we compute a sequence of progressively more precise bounds in a *cascade* [71], and only use more expensive estimators when necessary. We first describe a series of bounds relevant to the moments sketch in Section 5.1 and then show how they can be used in end-to-end queries in Section 5.2.

5.1 Moment-based inequalities

Given the statistics in a moments sketch, we apply a variety of classical inequalities to derive bounds on the quantiles. These provide some worst-case error guarantees on quantile estimates which we can report to users along with the quantile estimate, and are also essential in allowing fast query processing on threshold queries over multiple groups.

One simple inequality we make use of is Markov's inequality. Given a non-negative dataset D with moments μ_i Markov's inequality tells us that for any value x, $\text{rank}(x) \geq n\left(1 - \frac{\mu_k}{t^k}\right)$ where the rank is the number of elements in D less than x. We apply this to arbitrary datasets using the transformation $D' = \{x - x_{\min} : x \in D\}$ since one can compute the moments μ'_i of D' using the binomial formula. Similarly we can use the transformations $D^- = \{x_{\max} - x : x \in D\}$ and $D^l = \{\log(x) : x \in D\}$ to obtain lower bounds and analogous bounds on the log moments.

We apply Markov's inequality to a quantile estimate $x = q_\phi$ on each of the transformed datasets and for each moment μ_k above to obtain upper or lower bounds on the rank $r(t)$ and the take tightest

Algorithm 2: Threshold Query Cascade

```
macro CheckBound($r_{\text{lower}}, r_{\text{upper}}, r_t$)
    if $r_{\text{lower}} > r_t$ then
        return true
    else if $r_{\text{upper}} < r_t$ then
        return false
function Threshold(threshold $t$, quantile $\phi$)
    if $t > x_{\max}$ then
        return false
    if $t < x_{\min}$ then
        return true
    $r_{\text{lower}}, r_{\text{upper}} \leftarrow$ MarkovBound($t$)        ▷ Markov Bound
    CheckBound($r_{\text{lower}}, r_{\text{upper}}, n\phi$)
    $r_{\text{lower}}, r_{\text{upper}} \leftarrow$ RTTBound($t$)        ▷ RTT Bound
    CheckBound($r_{\text{lower}}, r_{\text{upper}}, n\phi$)
    $q_\phi \leftarrow$ MaxEntQuantile($\phi$)        ▷ Maximum Entropy
    return $q_\phi > t$
```

upper and lower bound. These upper and lower bounds on the rank then provide an upper bound on the quantile error ϵ for a ϕ-quantile estimate. We refer to this procedure as the MarkovBound procedure.

The authors in [63] provide a procedure for computing tighter but more expensive bounds on the CDF $F(t)$ of a distribution given its moments. We refer to this procedure as the RTTBound procedure, and as with the Markov bounds, use it to bound the rank of a computed quantile estimate q_ϕ. The RTTBound procedure does not make use of the standard moments and log moments simultaneously. When applying RTTBound to the moments sketch, we run the RTTBound procedure once on the standard moments and once on log moments and take the tighter of the bounds.

5.2 Cascades for Threshold queries

Given a moments sketch, Algorithm 2 shows how we calculate `Threshold`(t, ϕ): whether the dataset has quantile estimate q_ϕ above a fixed cutoff t. We use this routine whenever we answer queries on groups with a predicate $q_\phi > t$, allowing us to check whether a subgroup should be included in the results without computing the quantile estimate directly. The threshold check routine first performs a simple filter on whether the threshold t falls in the range $[x_{\min}, x_{\max}]$. Then, we can use the Markov inequalities and the RTTBound routine to calculate lower and upper bounds on the rank of the threshold rank(t) in the subpopulation. These bounds are used to determine if we can resolve the threshold predicate immediately. If not, we finally use our entropy optimization routine to estimate q_ϕ.

The Markov and RTTBound bounds are cheaper to compute than our maximum entropy estimate, making threshold predicates cheaper to evaluate than explicit quantile estimates. The bounds apply to any distribution or dataset that matches the moments in a moments sketch, so this routine has no false negatives and is consistent with calculating the maximum entropy quantile estimate up front.

6 EVALUATION

In this section we evaluate the efficiency and accuracy of the moments sketch in a series of microbenchmarks, and then show how the moments sketch provides end-to-end performance improvements in the Druid and Macrobase data analytics engines [9, 77].

This evaluation demonstrates that:

(1) The moments sketch supports 35 to 200× faster query times than comparably accurate summaries on quantile aggregations.

(2) The moments sketch provides $\epsilon_{\text{avg}} \leq 0.01$ estimates across a range of real-world datasets using less than 200 bytes of storage.

(3) Maximum entropy estimation is more accurate than alternative moment-based quantile estimates, and our solver improves estimation time by 200× over naive solutions.

(4) Integrating the moments sketch as a user-defined sketch provides 7× faster quantile queries than the default quantile summary in Druid workloads.

(5) Applying a cascade of bounds to moments sketch estimation provides 25× higher query throughput compared to direct moments sketch usage in Macrobase threshold queries .

Throughout the evaluations, the moments sketch is able to accelerate a variety of workloads when the number of summary aggregations exceeds the number of queries, when space is at a premium, and when interactive sub-100ms query response times are essential.

6.1 Experimental Setup

We implement the moments sketch and its quantile estimation routines in Java[2]. This allows for direct comparisons with the open source quantile summaries in the Java-based Yahoo! Data Sketches library [1], as well as those in Spark-SQL [8] and Druid [77], and also enables integration with the Java-based Druid [77] and MacroBase [9] systems. In our experimental results, we use the abbreviation M-Sketch to refer to the moments sketch.

We compare against a number of alternative quantile summaries: our implementation of a mergeable equi-width histogram (`EW-Hist`), the Greenwald-Khanna [33] (`GK`) sketch as implemented in Spark-SQL [8], the AVL-tree T-Digest [28] sketch (`T-Digest`), the streaming histogram (`S-Hist`) in [13] as implemented in Druid, reservoir sampling [72] (`Sampling`), and the low-discrepancy mergeable sketch (`MS2012`) from [3], both implemented in the Yahoo! datasketches library [1]. We implemented the `EW-Hist` as a mergeable sketch as in [62] by using power-of-two ranges and re-scaling on merge. Each quantile summary has a *size parameter* controlling its memory usage, which we will vary in our experiments. Our implementations and benchmarks use double precision floating point values. During moments sketch quantile estimation we run Newton's method until the moments match to within $\delta = 10^{-9}$, and select k_1, k_2 using a maximum condition number $\kappa_{\max} = 10^4$.

We quantify the accuracy of a quantile estimate using the quantile error ε as defined in Section 3.1. Then, as in [48] we can compare the accuracies of summaries on a given dataset by computing their

[2]https://github.com/stanford-futuredata/msketch

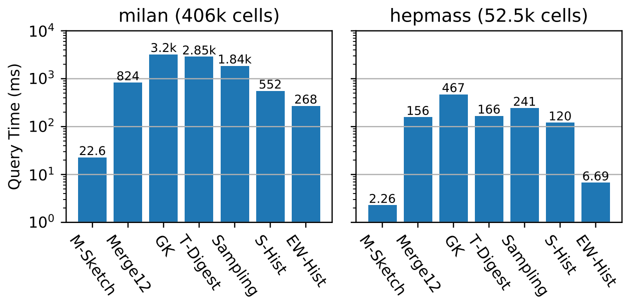

Figure 3: Total query time on $\epsilon_{\text{avg}} \leq .01$ quantile aggregation queries. The moments sketch enables significantly faster queries at this accuracy.

average error ϵ_{avg} over a set of uniformly spaced ϕ-quantiles. In the evaluation that follows, we test on 21 equally spaced ϕ between 0.01 and 0.99.

We evaluate each summary via single-threaded experiments on a machine with an Intel Xeon E5-4657L 2.40GHz processor and 1TB of RAM, omitting the time to load data from disk.

6.1.1 Datasets. We make use of six real-valued datasets in our experiments: The *milan* dataset consists of 81 million Internet usage measurements from Nov. 2013 in the Telecom Italia Call Data Records [38]. The *hepmass* dataset consists of 10.5 million values corresponding to the first feature in the UCI [46] HEPMASS dataset. The *occupancy* dataset consists of 20 thousand CO2 measurements from the UCI Occupancy Detection dataset. The *retail* dataset consists of 530 thousand positive purchase quantities from the UCI Online Retail dataset. The *power* dataset consists of 2 million Global Active Power measurements from the UCI Individual Household Electric Power Consumption dataset. The *exponential* dataset consists of 100 million synthetic values from an exponential distribution with $\lambda = 1$.

6.2 Performance Benchmarks

We begin with a series of microbenchmarks evaluating the moments sketch query times and accuracy.

6.2.1 Query Time. Our primary metric for evaluating the moments sketch is total query processing time. We evaluate quantile query times over the cube aggregation scenarios described in Section 1, by dividing our datasets into cells of 200 values – corresponding to pre-aggregated cube data – and maintaining quantile summaries for each cell. In this performance microbenchmark, the cells are grouped based on their sequence in the dataset: in Section 7 we will examine more realistic groupings. Then we measure the time it takes to calculate a quantile estimate for the entire dataset by performing a streaming sequence of merges.

Figure 3 shows the total query time to merge the summaries and then compute a quantile estimate when each summary is instantiated at the smallest size sufficient to achieve $\epsilon_{\text{avg}} \leq .01$ accuracy. We provide the parameters we used and average observed space usage in Table 1. On the long-tailed milan dataset, the `S-Hist` and `EW-Hist` summaries are unable to achieve $\epsilon_{\text{avg}} \leq .01$ accuracy with less than 100 thousand buckets, so we provide timings at 100 buckets for comparison – their query times are worse with more buckets.

dataset sketch	milan param	size (b)	hepmass param	size (b)
`M-Sketch`	order $k = 10$	200	$k = 3$	72
`MS2012`	size $k = 32$	5920	$k = 32$	5150
`GK`	error $\epsilon = \frac{1}{50}$	592	$\epsilon = \frac{1}{50}$	608
`T-Digest`	$\delta = 5.0$	769	$\delta = 1.5$	93
`Sampling`	1000 samples	8010	1000	8010
`S-Hist`	100 bins	1220	100	1220
`EW-Hist`	100 bins	812	15	132

Table 1: Summary size parameters used in Figure 3. We use these parameters to compare the query times at $\epsilon_{\text{avg}} \leq .01$ accuracy.

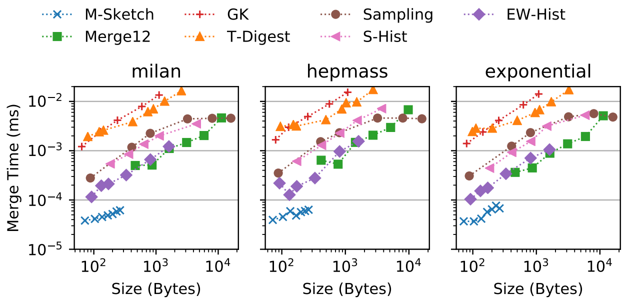

Figure 4: Per-merge latencies. The moments sketch provides faster merge times than alternative summaries at the same size.

The moments sketch is the only accurate summary which provides sub-100ms query times on both datasets, and provides 35 to 70× faster query times than `MS2012`, the next fastest accurate summary.

6.2.2 Merge and Estimation Time. Recall that for a basic aggregate quantile query $t_{\text{query}} = t_{\text{merge}} \cdot n_{\text{merge}} + t_{\text{est}}$. Thus we also measure t_{merge} and t_{est} to quantify the regimes where the moments sketch performs well. In these experiments, we vary the summary size parameters, though many summaries have a minimum size, and the moments sketch runs into floating point precision issues on some datasets for $k \geq 15$ so we do not include results for larger k (see Figure 9).

In Figure 4 we evaluate the average time required to merge one of the cell summaries. Larger summaries are more expensive to merge, and the moments sketch has faster (< 50ns) merge times throughout its size range. When comparing summaries using the parameters in Table 1, the moments sketch has up to 200x faster merge times than other summaries with the same accuracy.

The other major contributor to query time is estimation time. In Figure 5 we measure the time to estimate quantiles given an existing summary. The moments sketch provides on average 2 ms estimation times, though estimation time can be higher for small k when our estimator chooses higher k_1, k_2 to achieve better accuracy. This is the cause for the spike at $k = 4$ in the milan dataset and users can can mitigate this by lowering the condition number threshold $\kappa_{\max}$. Other summaries support microsecond estimation times. The moments sketch thus offers a tradeoff of better merge time for worse estimation time. If users require faster estimation times, the

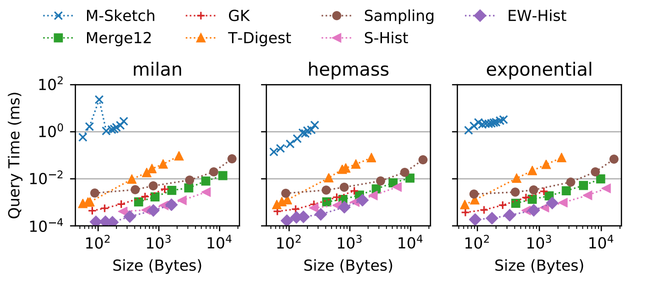

Figure 5: Quantile Estimation time. Estimation time on the moments sketch is slower than other sketches but under 3ms for $k = 10$.

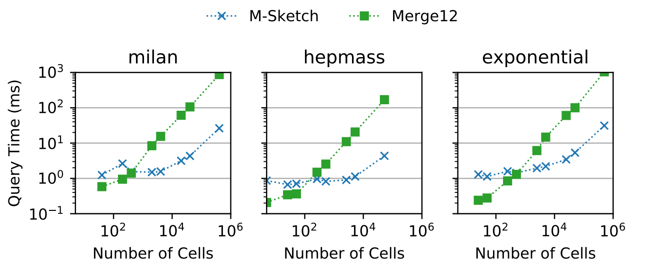

Figure 6: Varying the number of merges for a moments sketch with $k = 10$ and MS2012 with $k = 32$. Merge time dominates moments sketch query time for $n_{\text{merge}} \geq 10^4$.

cascades in Section 5.2 and the alternative estimators in Section 6.3 can assist.

We show how the merge time and estimation time tradeoff define regimes where each component dominates depending on the number of merges. In Figure 6 we measure how the query time changes for queries requiring different numbers of merges over pre-aggregated summaries on cells of 200 rows. We use the moments sketch with $k = 10$ and compare against the $k = 32$ MS2012 sketch as a baseline. Since the query time for the moments sketch is in milliseconds, merge time begins to dominate for $n_{\text{merge}} \geq 10^4$ and beyond that the moments sketch provides better performance than the MS2012 sketch. However, the moments sketch estimation times dominate when $n_{\text{merge}} \leq 100$.

6.2.3 Accuracy. The moments sketch accuracy is dataset dependent, so in this section we compare the average quantile error on our evaluation datasets.

Figure 7 illustrates the average quantile error ϵ_{avg} for summaries of different sizes. The moments sketch achieves $\epsilon \leq 10^{-4}$ accuracy on the synthetic exponential dataset, and $\epsilon \leq 10^{-3}$ accuracy on the high entropy hepmass dataset. On other datasets it is able to achieve $\epsilon_{\text{avg}} \leq 0.01$ with less than 200 bytes of space. The EW-Hist summary, while efficient to merge, provides less accurate estimates than the moments sketch and low accuracy in the long-tailed milan and retail datasets. The moments sketch offers a unique combination of fast merge times, small space usage, and accuracy on real-world datasets.

For comparison, Figure 8 shows the average guaranteed upper bound error provided by different summaries. These are in general higher than the observed errors. We use the RTTBound routine in

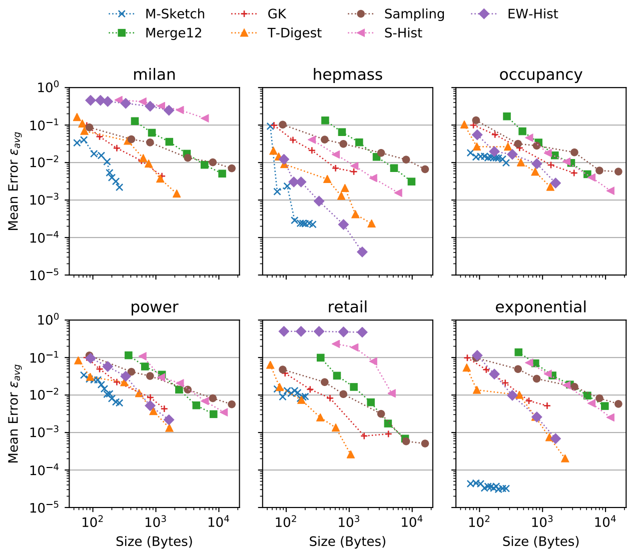

Figure 7: Average error for summaries of different sizes. The moments sketch delivers consistent $\epsilon_{\text{avg}} \leq 0.015$ with less than 200 bytes.

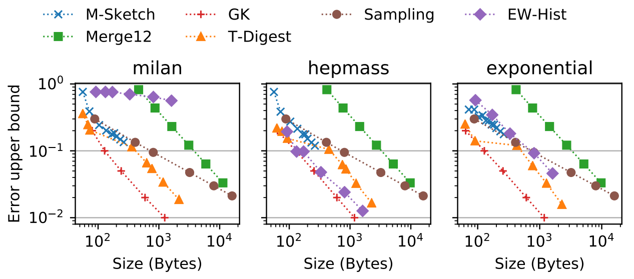

Figure 8: Average bound size for summaries of different sizes. No summary is able to provide $\epsilon_{bound} \leq .01$ guarantees with less than 1000 bytes.

Section 5.1 to bound the moments sketch error. We omit the S-Hist since it does not provide upper bounds.

We do not evaluate the moments sketch for $k \geq 15$ since we use Chebyshev polynomials as described in Section 4.3 during maximum entropy estimation, and the precision loss due to calculating Chebyshev polynomials from the moments sketch grows with higher k. Without converting to Chebyshev polynomials, the condition number of the optimization problem would grow even faster so we are limited to $k \leq 15$. Figure 9 shows the precision loss during Chebyshev polynomial calculation $\Delta\mu = |\mu_i - \hat{\mu}_i|$ where μ_i is the true Chebyshev moment and $\hat{\mu}_i$ is the value calculated from the moments sketch. Precision loss is severe on datasets such as the occupancy dataset for $k \geq 20$.

Unlike the EW-Hist summary, the moments sketch can remain accurate in the presence of very large values in a dataset. In Figure 10 we evaluate the effect of adding a fixed fraction $\delta = 0.01$ of outlier values from a Gaussian with mean μ_o and standard deviation $\sigma = 0.1$ to a dataset of 10 million standard Gaussian points. As we increase the magnitude μ_o of the outliers, the EW-Hist summaries

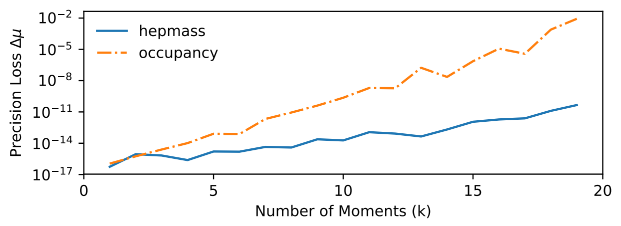

Figure 9: Precision loss from converting powers to Chebyshev polynomials. On the occupancy dataset, the Chebyshev polynomials are inaccurate past $k \geq 20$.

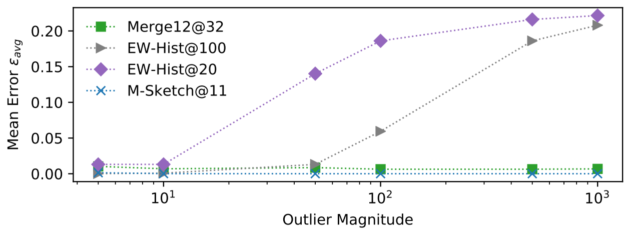

Figure 10: Mean error on a Gaussian dataset with outliers of different magnitudes added. The moments sketch remains accurate for large outliers, but the `EW-Hist` accuracy degrades.

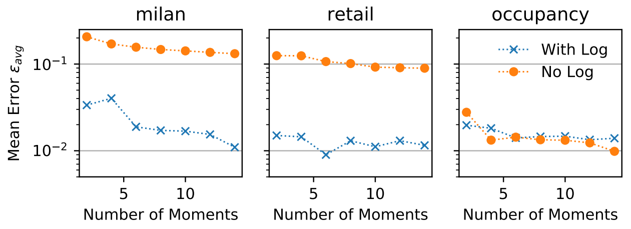

Figure 11: Accuracy with and without log moments. Given the same total space budget, log moments improve accuracy on the long-tailed milan and retail datasets, and do not affect accuracy significantly on other datasets such as occupancy

with 20 and 100 bins lose accuracy though a moments sketch with $k = 11$ remains accurate. The `MS2012` sketch is agnostic to value magnitudes and is unaffected by the outliers. If extremely large outliers are expected, floating point precision suffers and the moments sketch can be used in conjunction with standard outlier removal techniques.

6.3 Quantile Estimation Lesion Study

To evaluate each component of our quantile estimator design, we compared the accuracy and estimation time of a variety of alternative estimators on the milan and hepmass datasets. We evaluate the impact of using log moments, the maximum entropy distribution, and our optimizations to estimation.

To examine effectiveness of log moments, we compare our maximum entropy quantile estimator accuracy with and without log moments. For a fair comparison, we compare the estimates produced from k standard moments and no log moments with those produced from up to $\frac{k}{2}$ of each. Figure 11 illustrates how on some long-tailed datasets, notably milan and retail, log moments reduce the error from $\epsilon > .15$ to $\epsilon < .015$, given the same total amount of moments. On other datasets, log moments do not have a significant impact, but we include them in the moments sketch for merges with populations that will benefit from log moments.

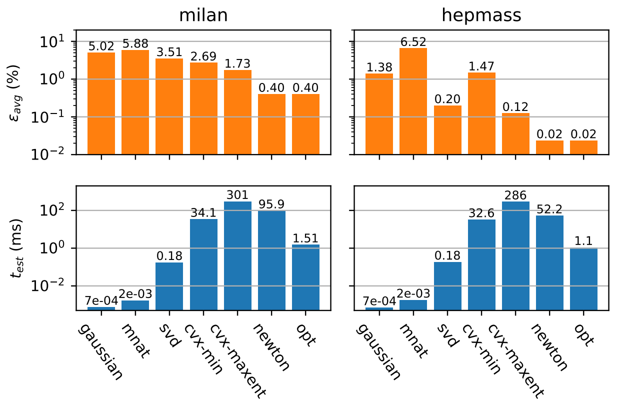

Figure 12: Lesion study comparing our optimized maximum entropy solver to other estimators. Our `opt` estimator provides at least $5\times$ less error than estimators that do not use maximum entropy, and up to $200\times$ faster estimation times than naive maximum entropy solvers.

We compare our estimator with a number of other estimators that make use of the same moments. Some of these estimators do not make use of the maximum entropy principle. The gaussian estimator fits a Gaussian distribution to the mean and standard deviation. The mnat estimator uses the closed form discrete CDF estimator in [54]. The svd estimator discretizes the domain and uses singular value decomposition to solve for a distribution with matching moments. The cvx-min estimator also discretizes the domain and uses a convex solver a distribution with minimal maximum density and matching moments. Other estimators solve for the maximum entropy principle using alternative techniques. The cvx-maxent estimator discretizes the domain and uses a convex solver to maximize the entropy and match the moments, as described in Chapter 7 in [16]. The newton estimator implements our estimator without the integration techniques in Sec. 4.3, and uses adaptive Romberg integration instead [61].

Figure 12 illustrates the average quantile error and estimation time for these estimators. We run these experiments with $k = 10$ moments. For uniform comparisons with other estimators, on the milan dataset we only use the log moments, and on the hepmass dataset we only use the standard moments. We perform all discretizations using 1000 uniformly spaced points, and we use the ECOS convex solver [27]. Our maximum entropy solver provides at least $5\times$ less error than estimators that do not use maximum entropy, and better accuracy than estimating the maximum entropy using a discretized grid. Furthermore, our optimizations are able to improve the estimation time by a factor of up to $200\times$ over an implementation using generic solvers.

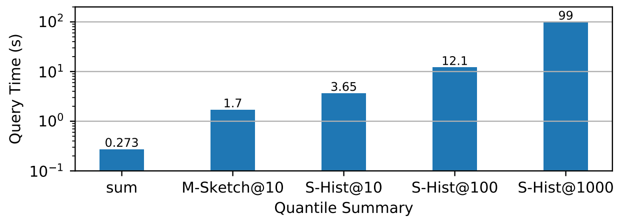

Figure 13: Druid end-to-end query benchmark. The moments sketch allows for faster query times than the comparable S-Hist summary with 100 bins. Runtime for a native sum operation is a lower bound on query time.

7 APPLYING THE MOMENTS SKETCH

In this section, we evaluate how the moments sketch merge and quantile estimation routines affect performance in larger data systems. In particular, we will examine how the moments sketch can improve end-to-end query performance in the Druid analytics engine, as part of a cascade in the Macrobase feature selection engine [9], and as part of exploratory sliding window queries. Finally, we will measure how moments sketch space usage can be further reduced in space-constrained settings using reduced precision floating point representations.

7.1 Druid Integration

To illustrate the utility of the moments sketch in a modern analytics engine, we integrate the moments sketch with Druid [77]. We do this by implementing moments sketch as an aggregation function extension, and compare the total query time on quantile queries of the moments sketch with the default S-Hist summary used in Druid and introduced in [13]. The authors in [13] observe on average 5% error for an S-Hist with 100 centroids, so we benchmark a moments sketch with $k = 10$ against S-Hists with 10, 100, and 1000 centroids.

In our experiments, we deployed Druid on a single node – the same machine described in section 6.1 – with the same base configuration used in the default Druid quickstart. In particular, this configuration dedicates 2 threads to process aggregations. Then, we ingest 26 million entries from the milan dataset at a one hour granularity and construct a cube over the grid ID and country dimensions, resulting in 10 million cells.

Figure 13 compares the total query time using the different summaries. The moments sketch provides 7× lower query times than a S-Hist with 100 bins. Furthermore, as discussed in Section 6.2.1, any S-Hist with fewer than 10 thousand buckets provides worse accuracy on milan data than the moments sketch. As a best-case baseline, we also show the time taken to compute a native sum query on the same data. The 1 ms cost of solving for quantile estimates from the moments sketch on this dataset is negligible here.

7.2 Threshold queries

In this section we evaluate how well the moments sketch and the cascades described in Section 5.2 improve performance when processing queries with threshold predicates. First we show in Section 7.2.1 how data exploration engines such as MacroBase benefit from using the moments sketch to search for anomalous dimension values. Then, we show in Section 7.2.2 how historical analytics queries can use the moments sketch to search and alert on sliding windows.

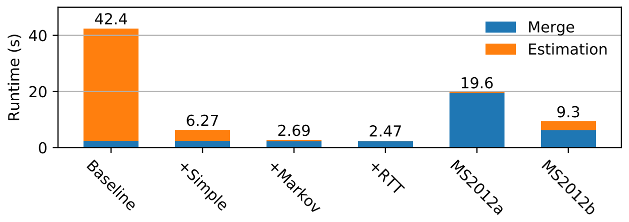

Figure 14: Runtime of MacroBase queries: the final moments sketch cascade outperforms queries using alternate sketches.

7.2.1 MacroBase Integration.

The MacroBase engine searches for dimension values with unusually high outlier rates in a dataset [9]. For example, given an overall 2% outlier rate, MacroBase may report when a specific app version has an outlier rate of 20%. We integrate the moments sketch with a simplified deployment of MacroBase where all values greater than the global 99th percentile t_{99} are considered outliers. We then query MacroBase for all dimension values with outlier rate at least $r = 30\times$ greater than the overall outlier rate. This is equivalent to finding subpopulations whose 70th percentile is greater than t_{99}.

Given a cube with pre-aggregated moments sketches for each dimension value combination and no materialized roll-ups, MacroBase merges the moments sketches to calculate the global t_{99}, and then runs Algorithm 2 on every dimension-value subpopulation, searching for subgroups with $q_{.7} > t_{99}$. We evaluate the performance of this query on 80 million rows of the milan internet usage data from November 2013, pre-aggregated by grid ID, country, and at a four hour granularity. This resulted in 13 million cube cells, each with its own moments sketch.

Running the MacroBase query produces 19 candidate dimension values. We compare the total time to process this query using direct quantile estimates, our cascades, and the alternative MS2012 quantile sketch. In the first approach (MS2012a), we merge summaries during MacroBase execution as we do with a moments sketch. In the second approach (MS2012b), we take advantage of the fast query time of the MS2012 sketch to calculate the number of values greater than the t_{99} for each dimension value combination – each cube cell – so that MacroBase can accumulate these counts directly, instead of the sketches. This approach incurs a higher query time but avoids much of the cost of merging. We present this as an optimistic baseline, so it is not always a feasible substitute for merging summaries.

Figure 14 shows the query times for these different methods: the baseline method calculates quantile estimates directly for every threshold, we show the effect of incrementally adding each stage of our cascade ending with +RTTBound, and we also compare against the two usages of the MS2012 sketch. Each successive stage of the cascade improves query time substantially. With the complete cascade, estimation time is negligible compared to merge time. Furthermore, the moments sketch with cascades has 7.9× lower

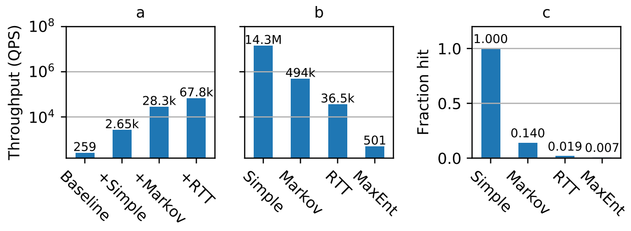

Figure 15: Cascades in MacroBase: (a) as we incrementally add cascade stages, threshold query throughput increases. (b) The cascade proceeds from faster to slower estimates. (c) Each stage of the cascade processes a smaller fraction of queries.

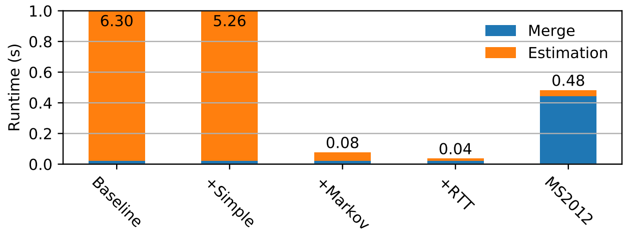

Figure 16: Sliding window query: moments sketch with cascades runs 13× faster than MS2012.

query times than using the MS2012 sketch, and even 3.7× lower query times than the MS2012b baseline.

In Figure 15 we examine the impact the cascade has on estimation time directly. Each additional cascade stage improves threshold query throughput and is more expensive than the last. The complete cascade is over 250× faster than this baseline, and 25× faster than just using a simple range check.

7.2.2 Sliding Window Queries. Threshold predicates are broadly applicable in data exploration queries. In this section, we evaluate how the moments sketch and our cascade improves performance on queries for sliding windows that exceed an alert threshold. This is useful when, for instance, users are searching for time windows of unusually high CPU usage spikes.

For this benchmark, we aggregated the 80 million rows of the milan dataset at a 10-minute granularity, which produced 4320 panes that spanned the month of November. We augmented the milan data with two spikes corresponding to hypothetical anomalies. Each spike spanned a two-hour time frame and contributed 10% more data to those time frames. Given a global 99th percentile of around 500 and a maximum value of 8000, we added spikes with values $x = 2000$ and $x = 1000$

We then queried for the 4-hour time windows whose 99th percentile was above a threshold $t = 1500$. When processing this query using a moments sketch, we can update sliding windows using turnstile semantics, subtracting the values from the oldest pane and merging in the new one.

Figure 16 shows the runtime of the sliding window query using both the moments sketch and MS2012. The cascade improves the estimation time of the moments sketch by filtering out windows that cannot pass the threshold. Faster moments sketch merge times and the use of turnstile semantics then allow for 13× faster queries than MS2012.

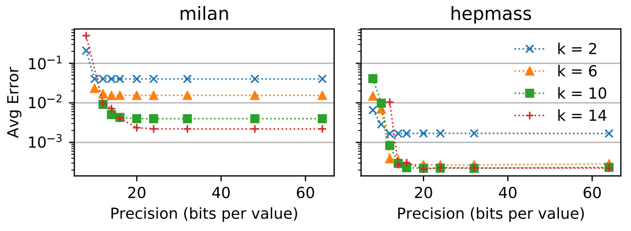

Figure 17: Average error for low-precision moments sketches after 100 thousand merges. Twenty bits of precision is sufficient to maintain accuracy for both datasets.

7.3 Low-precision storage

When implemented using double precision floating point, the moments sketch already has lower space overheads than alternative summaries at the same accuracy (Figure 7 in Section 6.2.3). In settings where space is heavily constrained, the moments sketch can be compressed further by reducing the precision of the sketch contents using randomized rounding.

As a preliminary proof-of-concept of this approach, we created an encoder that compresses the double precision floating point values in a moments sketch using reduced floating point precision, quantizing the significand and removing unused bits in the exponent. This low-precision representation has a negligible impact on merge times since we can convert them to and from native double precision using simple bit manipulation.

We then evaluate the encoding by constructing 100 thousand pre-aggregated moments sketches, reducing their precision, and then merging them and querying for quantiles on the aggregation. Figure 17 illustrates how as we reduce the number of bits used, the quality of the final estimate remains stable until we reach a minimum threshold, after which further reduction decreases the accuracy of the quantile estimates. On the milan dataset, a moments sketch with $k = 10$ can be stored with 20 bits per value without noticeably affecting our quantile estimates, representing a 3× space reduction compared to standard double precision floating point.

8 CONCLUSION

In this paper, we show how to improve the performance of quantile aggregation queries using statistical moments. Low merge overhead allows the moments sketch to outperform comparably accurate existing summaries when queries aggregate more than 10 thousand summaries. By making use of the method of moments and the maximum entropy principle, the moments sketch provides $\epsilon_{\text{avg}} \leq 0.01$ accuracy on real-world datasets, while the use of numeric optimizations and cascades keep query times at interactive latencies.

REFERENCES

[1] 2017. Yahoo! Data Sketches Library. (2017). https://datasketches.github.io/.
[2] Lior Abraham, John Allen, Oleksandr Barykin, Vinayak Borkar, Bhuwan Chopra, Ciprian Gerea, Daniel Merl, Josh Metzler, David Reiss, Subbu Subramanian, Janet L. Wiener, and Okay Zed. 2013. Scuba: Diving into Data at Facebook. *VLDB* 6, 11 (2013), 1057–1067.
[3] Pankaj K. Agarwal, Graham Cormode, Zengfeng Huang, Jeff Phillips, Zhewei Wei, and Ke Yi. 2012. Mergeable Summaries. In *PODS*.

[4] Sameer Agarwal, Barzan Mozafari, Aurojit Panda, Henry Milner, Samuel Madden, and Ion Stoica. 2013. BlinkDB: Queries with Bounded Errors and Bounded Response Times on Very Large Data. In *EuroSys*. 29–42. https://doi.org/10.1145/2465351.2465355

[5] N.I. Akhiezer. 1965. *The Classical Moment Problem and Some Related Questions in Analysis*. Oliver & Boyd.

[6] Luigi Ambrosio, Nicola Gigli, and Giuseppe Savaré. 2008. *Gradient flows: in metric spaces and in the space of probability measures*. Springer Science & Business Media.

[7] Animashree Anandkumar, Daniel Hsu, and Sham M Kakade. 2012. A method of moments for mixture models and hidden Markov models. In *COLT*. 33–1.

[8] Michael Armbrust, Reynold S Xin, Cheng Lian, Yin Huai, Davies Liu, Joseph K Bradley, Xiangrui Meng, Tomer Kaftan, Michael J Franklin, Ali Ghodsi, et al. 2015. Spark sql: Relational data processing in spark. In *SIGMOD*. 1383–1394.

[9] Peter Bailis, Edward Gan, Samuel Madden, Deepak Narayanan, Kexin Rong, and Sahaana Suri. 2017. MacroBase: Prioritizing attention in fast data. In *SIGMOD*. 541–556.

[10] A. Balestrino, A. Caiti, A. Noe', and F. Parenti. 2003. Maximum entropy based numerical algorithms for approximation of probability density functions. In *2003 European Control Conference (ECC)*. 796–801.

[11] K. Bandyopadhyay, A. K. Bhattacharya, Parthapratim Biswas, and D. A. Drabold. 2005. Maximum entropy and the problem of moments: A stable algorithm. *Phys. Rev. E* 71 (May 2005), 057701. Issue 5. https://doi.org/10.1103/PhysRevE.71.057701

[12] Mikhail Belkin and Kaushik Sinha. 2010. Polynomial Learning of Distribution Families. In *FOCS*. 10.

[13] Yael Ben-Haim and Elad Tom-Tov. 2010. A streaming parallel decision tree algorithm. *Journal of Machine Learning Research* 11, Feb (2010), 849–872.

[14] Adam L Berger, Vincent J Della Pietra, and Stephen A Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational linguistics* 22, 1 (1996), 39–71.

[15] B. Beyer, C. Jones, J. Petoff, and N.R. Murphy. 2016. *Site Reliability Engineering: How Google Runs Production Systems*. O'Reilly Media, Incorporated. https://books.google.com/books?id=81UrjwEACAAJ

[16] Stephen Boyd and Lieven Vandenberghe. 2004. *Convex Optimization*. Cambridge University Press, New York, NY, USA.

[17] Lucas Braun, Thomas Etter, Georgios Gasparis, Martin Kaufmann, Donald Kossmann, Daniel Widmer, Aharon Avitzur, Anthony Iliopoulos, Eliezer Levy, and Ning Liang. 2015. Analytics in Motion: High Performance Event-Processing AND Real-Time Analytics in the Same Database. In *SIGMOD*. 251–264. https://doi.org/10.1145/2723372.2742783

[18] Mihai Budiu, Rebecca Isaacs, Derek Murray, Gordon Plotkin, Paul Barham, Samer Al-Kiswany, Yazan Boshmaf, Qingzhou Luo, and Alexandr Andoni. 2016. Interacting with Large Distributed Datasets Using Sketch. In *Eurographics Symposium on Parallel Graphics and Visualization*.

[19] Chiranjeeb Buragohain and Subhash Suri. 2009. Quantiles on streams. In *Encyclopedia of Database Systems*. Springer, 2235–2240.

[20] Yixin Chen, Guozhu Dong, Jiawei Han, Jian Pei, Benjamin W Wah, and Jianyong Wang. 2006. Regression cubes with lossless compression and aggregation. *TKDE* 18, 12 (2006), 1585–1599.

[21] Graham Cormode and Minos Garofalakis. 2007. Streaming in a connected world: querying and tracking distributed data streams. In *SIGMOD*. 1178–1181.

[22] Graham Cormode, Minos Garofalakis, Peter J Haas, and Chris Jermaine. 2012. Synopses for massive data: Samples, histograms, wavelets, sketches. *Foundations and Trends in Databases* 4, 1–3 (2012), 1–294.

[23] Graham Cormode and S. Muthukrishnan. 2005. An Improved Data Stream Summary: The Count-min Sketch and Its Applications. *J. Algorithms* 55, 1 (April 2005), 58–75. https://doi.org/10.1016/j.jalgor.2003.12.001

[24] Chuck Cranor, Theodore Johnson, Oliver Spataschek, and Vladislav Shkapenyuk. 2003. Gigascope: A Stream Database for Network Applications. In *SIGMOD*. 647–651. https://doi.org/10.1145/872757.872838

[25] Mayur Datar, Aristides Gionis, Piotr Indyk, and Rajeev Motwani. 2002. Maintaining stream statistics over sliding windows. *SIAM journal on computing* 31, 6 (2002), 1794–1813.

[26] Jeffrey Dean and Luiz André Barroso. 2013. The Tail at Scale. *Commun. ACM* 56, 2 (2013), 74–80.

[27] A. Domahidi, E. Chu, and S. Boyd. 2013. ECOS: An SOCP solver for embedded systems. In *European Control Conference (ECC)*. 3071–3076.

[28] Ted Dunning and Otmar Ertl. 2017. Computing extremeley accurate quantiles using t-digests. https://github.com/tdunning/t-digest. (2017).

[29] W. Feng, C. Zhang, W. Zhang, J. Han, J. Wang, C. Aggarwal, and J. Huang. 2015. STREAMCUBE: Hierarchical spatio-temporal hashtag clustering for event exploration over the Twitter stream. In *ICDE*. 1561–1572. https://doi.org/10.1109/ICDE.2015.7113425

[30] Walter Gautschi. 1978. Questions of Numerical Condition Related to Polynomials. In *Recent Advances in Numerical Analysis*, Carl De Boor and Gene H. Golub (Eds.). Academic Press, 45 – 72. https://doi.org/10.1016/B978-0-12-208360-0.50008-5

[31] Walton C Gibson. 2014. *The method of moments in electromagnetics*. CRC press.

[32] Jim Gray, Surajit Chaudhuri, Adam Bosworth, Andrew Layman, Don Reichart, Murali Venkatrao, Frank Pellow, and Hamid Pirahesh. 1997. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *Data mining and knowledge discovery* 1, 1 (1997), 29–53.

[33] Michael Greenwald and Sanjeev Khanna. 2001. Space-efficient online computation of quantile summaries. In *SIGMOD*, Vol. 30. 58–66.

[34] Alex Hall, Alexandru Tudorica, Filip Buruiana, Reimar Hofmann, Silviu-Ionut Ganceanu, and Thomas Hofmann. 2016. Trading off Accuracy for Speed in PowerDrill. In *ICDE*. 2121–2132.

[35] Lars Peter Hansen. 1982. Large sample properties of generalized method of moments estimators. *Econometrica* (1982), 1029–1054.

[36] Daniel N. Hill, Houssam Nassif, Yi Liu, Anand Iyer, and S.V.N. Vishwanathan. 2017. An Efficient Bandit Algorithm for Realtime Multivariate Optimization. In *KDD*. 1813–1821. https://doi.org/10.1145/3097983.3098184

[37] Tim Hunter, Hossein Falaki, and Joseph Bradley. 2016. Approximate Algorithms in Apache Spark: HyperLogLog and Quantiles. https://databricks.com/blog/2016/05/19/approximate-algorithms-in-apache-spark-hyperloglog-and-quantiles.html. (2016).

[38] Telecom Italia. 2015. Telecommunications - SMS, Call, Internet - MI. (2015). https://doi.org/10.7910/DVN/EGZHFV http://dx.doi.org/10.7910/DVN/EGZHFV.

[39] E. T. Jaynes. 1957. Information Theory and Statistical Mechanics. *Phys. Rev.* 106 (May 1957), 620–630. Issue 4.

[40] Ramesh Johari, Pete Koomen, Leonid Pekelis, and David Walsh. 2017. Peeking at A/B Tests: Why It Matters, and What to Do About It. In *KDD*. 1517–1525. https://doi.org/10.1145/3097983.3097992

[41] Adam Tauman Kalai, Ankur Moitra, and Gregory Valiant. 2010. Efficiently learning mixtures of two Gaussians. In *STOC*. 553–562.

[42] N. Kamat, P. Jayachandran, K. Tunga, and A. Nandi. 2014. Distributed and interactive cube exploration. In *IDCE*. 472–483. https://doi.org/10.1109/ICDE.2014.6816674

[43] David Kempe, Alin Dobra, and Johannes Gehrke. 2003. Gossip-based computation of aggregate information. In *FOCS*. 482–491.

[44] Weihao Kong and Gregory Valiant. 2017. Spectrum estimation from samples. *Ann. Statist.* 45, 5 (10 2017), 2218–2247.

[45] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *ICML*. 282–289.

[46] M. Lichman. 2013. UCI Machine Learning Repository. (2013). http://archive.ics.uci.edu/ml

[47] Shaosu Liu, Bin Song, Sriharsha Gangam, Lawrence Lo, and Khaled Elmeleegy. 2016. Kodiak: Leveraging Materialized Views for Very Low-latency Analytics over High-dimensional Web-scale Data. *VLDB* 9, 13 (2016), 1269–1280. https://doi.org/10.14778/3007263.3007266

[48] Ge Luo, Lu Wang, Ke Yi, and Graham Cormode. 2016. Quantiles over data streams: experimental comparisons, new analyses, and further improvements. *VLDB* 25, 4 (2016), 449–472.

[49] Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. 2002. TAG: A Tiny AGgregation Service for Ad-hoc Sensor Networks. *OSDI*. https://doi.org/10.1145/844128.844142

[50] Amit Manjhi, Suman Nath, and Phillip B Gibbons. 2005. Tributaries and deltas: Efficient and robust aggregation in sensor network streams. In *SIGMOD*. 287–298.

[51] Volker Markl, Peter J Haas, Marcel Kutsch, Nimrod Megiddo, Utkarsh Srivastava, and Tam Minh Tran. 2007. Consistent selectivity estimation via maximum entropy. *VLDB* 16, 1 (2007), 55–76.

[52] John C Mason and David C Handscomb. 2002. *Chebyshev polynomials*. CRC Press.

[53] Lawrence R. Mead and N. Papanicolaou. 1984. Maximum entropy in the problem of moments. *J. Math. Phys.* 25, 8 (1984), 2404–2417.

[54] Robert M. Mnatsakanov. 2008. Hausdorff moment problem: Reconstruction of distributions. *Statistics & Probability Letters* 78, 12 (2008), 1612 – 1618.

[55] Dominik Moritz, Danyel Fisher, Bolin Ding, and Chi Wang. 2017. Trust, but Verify: Optimistic Visualizations of Approximate Queries for Exploring Big Data. In *CHI*. 2904–2915. https://doi.org/10.1145/3025453.3025456

[56] J. I. Munro and M. S. Paterson. 1980. Selection and sorting with limited storage. *Theoretical Computer Science* (1980), 315–323.

[57] S. Muthukrishnan. 2005. Data Streams: Algorithms and Applications. *Found. Trends Theor. Comput. Sci.* 1, 2 (Aug. 2005), 117–236. https://doi.org/10.1561/0400000002

[58] Arnab Nandi, Cong Yu, Philip Bohannon, and Raghu Ramakrishnan. 2011. Distributed cube materialization on holistic measures. In *ICDE*. IEEE, 183–194.

[59] Kamal Nigam. 1999. Using maximum entropy for text classification. In *IJCAI Workshop on Machine Learning for Information Filtering*. 61–67.

[60] Carlos Ordonez, Yiqun Zhang, and Wellington Cabrera. 2016. The Gamma matrix to summarize dense and sparse data sets for big data analytics. *TKDE* 28, 7 (2016), 1905–1918.

[61] William H Press. 2007. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press.

[62] Ariel Rabkin, Matvey Arye, Siddhartha Sen, Vivek S. Pai, and Michael J. Freedman. 2014. Aggregation and Degradation in JetStream: Streaming Analytics in the Wide Area. In *NSDI*. 275–288.

[63] Sandor Racz, Arpad Tari, and Miklos Telek. 2006. A moments based distribution bounding method. *Mathematical and Computer Modelling* 43, 11 (2006), 1367 – 1382.

[64] Nelson Ray. 2013. The Art of Approximating Distributions: Histograms and Quantiles at Scale. http://druid.io/blog/2013/09/12/the-art-of-approximating-distributions.html. (2013).

[65] Sunita Sarawagi. 2000. User-adaptive exploration of multidimensional data. In *VLDB*, Vol. 2000. 307–316.

[66] Jayavel Shanmugasundaram, Usama Fayyad, and Paul S Bradley. 1999. Compressed data cubes for olap aggregate query approximation on continuous dimensions. In *KDD*. 223–232.

[67] Nisheeth Shrivastava, Chiranjeeb Buragohain, Divyakant Agrawal, and Subhash Suri. 2004. Medians and beyond: new aggregation techniques for sensor networks. In *International conference on Embedded networked sensor systems*. 239–249.

[68] RN Silver and H Röder. 1997. Calculation of densities of states and spectral functions by Chebyshev recursion and maximum entropy. *Physical Review E* 56, 4 (1997), 4822.

[69] U. Srivastava, P. J. Haas, V. Markl, M. Kutsch, and T. M. Tran. 2006. ISOMER: Consistent Histogram Construction Using Query Feedback. In *ICDE*. 39–39. https://doi.org/10.1109/ICDE.2006.84

[70] Manasi Vartak, Sajjadur Rahman, Samuel Madden, Aditya Parameswaran, and Neoklis Polyzotis. 2015. SeeDB: Efficient Data-driven Visualization Recommendations to Support Visual Analytics. *VLDB* 8, 13 (2015), 2182–2193. https://doi.org/10.14778/2831360.2831371

[71] Paul Viola and Michael Jones. 2001. Rapid object detection using a boosted cascade of simple features. In *CVPR*, Vol. 1. IEEE, I–511–I–518.

[72] Jeffrey S. Vitter. 1985. Random Sampling with a Reservoir. *ACM Trans. Math. Softw.* 11, 1 (1985), 37–57. https://doi.org/10.1145/3147.3165

[73] Abdul Wasay, Xinding Wei, Niv Dayan, and Stratos Idreos. 2017. Data Canopy: Accelerating Exploratory Statistical Analysis. In *SIGMOD*. 557–572.

[74] Larry Wasserman. 2010. *All of Statistics: A Concise Course in Statistical Inference.* Springer Publishing Company, Incorporated.

[75] Ruibin Xi, Nan Lin, and Yixin Chen. 2009. Compression and aggregation for logistic regression analysis in data cubes. *TKDE* 21, 4 (2009), 479–492.

[76] X. Xie, X. Hao, T. B. Pedersen, P. Jin, and J. Chen. 2016. OLAP over probabilistic data cubes I: Aggregating, materializing, and querying. In *ICDE*. 799–810.

[77] Fangjin Yang, Eric Tschetter, Xavier Léauté, Nelson Ray, Gian Merlino, and Deep Ganguli. 2014. Druid: A Real-time Analytical Data Store. In *SIGMOD*. 157–168.

[78] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J Franklin, Scott Shenker, and Ion Stoica. 2012. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *NSDI*. 2–2.