Rapid Approximate Aggregation with Distribution-Sensitive Interval Guarantees
[Technical Report]

Stephen Macke[1,2] Maryam Aliakbarpour[3] Ilias Diakonikolas[4]
Aditya Parameswaran[2] Ronitt Rubinfeld[3]
[1]University of Illinois (UIUC) [2]UC Berkeley [3]MIT [4]University of Wisconsin, Madison
{smacke,adityagp}@berkeley.edu {maryama@,ronitt@csail.}mit.edu ilias.diakonikolas@gmail.com

ABSTRACT

Aggregating data is fundamental to data analytics, data exploration, and OLAP. Approximate query processing (AQP) techniques are often used to accelerate computation of aggregates using samples, for which confidence intervals (CIs) are widely used to quantify the associated error. CIs used in practice fall into two categories: techniques that are *tight but not correct*, i.e., they yield tight intervals but only offer asymptotic guarantees, making them unreliable, or techniques that are *correct but not tight*, i.e., they offer rigorous guarantees, but are overly conservative, leading to confidence intervals that are too loose to be useful. In this paper, we develop a CI technique that is both correct and tighter than traditional approaches. Starting from conservative CIs, we identify two issues they often face: *pessimistic mass allocation* (PMA) and *phantom outlier sensitivity* (PHOS). By developing a novel *range-trimming* technique for eliminating PHOS and pairing it with known CI techniques without PMA, we develop a technique for computing CIs with strong guarantees that requires fewer samples for the same width. We implement our techniques underneath a sampling-optimized in-memory column store and show how they accelerate queries involving aggregates on three real datasets with typical speedups on the order of 10× over both traditional AQP-with-guarantees and exact methods, all while obeying accuracy constraints.

1. INTRODUCTION

Primitives for aggregation like AVG, SUM, and COUNT are key to making sense of and drawing insights from large volumes of data, powering applications in OLAP, exploratory data analysis, and visual analytics. Accelerating their computation is therefore of great importance. Approximate Query Processing (AQP) is commonly used to accelerate computation of these aggregates by estimating them on a subset or sample of the full data. Reasoning about the error of the estimates as introduced by approximation is crucial: consumers of approximate answers—ranging from human decision makers to automated processes—rely on confidence intervals (CIs) or error bounds as the foundation for understanding the quality of the approximate answer. Therefore, many AQP techniques come with CIs to allow for more confident or informed decisions made using approximate estimates.

Error bounding, or CI computation techniques take a confidence parameter $\delta \in [0, 1]$, with the semantics that the returned intervals $[g_\ell, g_r]$ fail to enclose the true aggregate $g^\star$ at most δ of the time. One can tune δ to be as small as needed at the cost of requiring more samples to achieve the same interval width $(g_r - g_\ell)$. Likewise, for a given δ, taking more samples typically causes the error bounding procedure to return a narrower confidence interval. Since δ is typically small, we use the phrase "with high probability" (w.h.p.) as shorthand for "with probability greater than $(1-\delta)$". CI computation techniques need to satisfy two goals: **(i) compactness:** *by minimizing the interval width* $g_r - g_\ell$, and **(ii) correctness:** *by ensuring that* $g^\star \in [g_\ell, g_r]$ *with high probability.* However, achieving both compactness and correctness is difficult.

```
SELECT Origin, AVG(DepDelay) FROM flights
GROUP BY Origin HAVING AVG(DepDelay) < 0
```

Figure 1: Origin airports with negative average delay. In this query, the AVG aggregates are consumed both by the user and by the system.

Existing techniques either prefer compactness over correctness (*asymptotic* techniques) or vice versa (*conservative* techniques):

Compactness without Correctness. *Asymptotic* error bounding techniques such as bootstrap CIs [29, 28, 77] or central limit theorem (CLT)-based CIs [67, 39] make assumptions about the distribution taken by the data given a "large enough" sample size. These procedures typically give CIs that are much tighter (and therefore more useful for drawing inferences about the query results), and have enjoyed numerous applications in database and visual analytics systems [62, 58, 59, 50, 32, 47], including Aqua [7], BlinkDB [10, 8], DBO [44], and online aggregation [40], and have furthermore seen a number of DBMS-specific extensions [77, 61].

However, these asymptotic techniques result in intervals that only enclose the true aggregate w.h.p. in the limit as the size of the sample grows to infinity[1]; i.e., they provide no real guarantees for any given finite instance, potentially leading to failures downstream. For example, consider the query in Figure 1, which determines origin airports whose departing flights are ahead-of-schedule, on average. An AQP system could use CIs to facilitate early stopping by using them to infer on which side of the HAVING threshold the various groups appear. If such a system relies on asymptotic CIs, it is prone to serious types of error, called *subset error* and *superset error* [58] , whereby certain tuples may be missing, and other tuples may appear spuriously.

Correctness without Compactness. Recognizing the downsides of asymptotic approaches, recent work [25, 11, 45, 65, 56] has begun to adopt *conservative* error bounders, which leverage concentration inequalities to compute CIs. These procedures return bounds that follow *probably approximately correct* (PAC) [69] semantics: given $\delta \in [0, 1]$, the probability that the procedure returns lower and upper bounds $[g_\ell, g_r]$ around the approximate aggregate $\hat{g}$ that fail to enclose the true aggregate $g^\star$ should be *at most* δ for *any* sample size (in contrast with asymptotic techniques, for which the probability converges to δ given a large enough sample). These techniques have

[1] The error of CLT-based methods shrinks as $\mathcal{O}\left(1/\sqrt{m}\right)$, but with constants depending on unknowns such as the third absolute normalized moment, according to the Berry-Esseen theorem [16, 30].

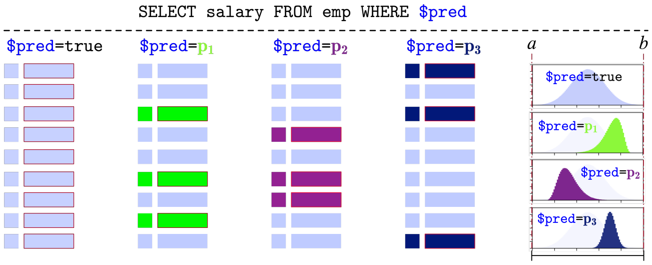


Figure 2: Few points may lie near the range bounds a and b, and with filters applied, the true range could be significantly smaller than $(b-a)$.

been used in online aggregation [40, 36] and more recently in work on visual analytics [11, 45, 65, 56].

In general, conservative methods such as those based on Hoeffding's inequality [41] or on the Hoeffding-Serfling inequality [66] rely on a-priori knowledge of *range bounds* a and b between which the data fall (typically inferred during data loading). Although they achieve the correctness goal of error bounders, when used for AVG, the CI width for Hoeffding-based error bounders scales with the range size $(b-a)$, creating at least two major issues in the context of a relational database, illustrated in Figure 2. ***(i) First, the presence of a very few outliers can significantly widen the range*** $[a,b]$ (and therefore the CI width), even though most of the data may lie in a much smaller range. In Figure 2, we see that even though the range of `salaries` is $b-a$ when `$pred = true`, most of the data is concentrated in the center of the range. ***(ii) Second, predicates and groupings may be applied during data exploration, so that the filtered data lies in a smaller range than*** $[a,b]$; in Figure 2, with `$pred =` $\mathbf{p_i}$, we see that the range of filtered salaries is much smaller than even the `$pred = true` case. However, direct application of Hoeffding-based methods do not account for the tighter range of the filtered data, instead treating the sampled tuples as if they were taken from the original (unfiltered) data.

Key Research Challenges and Contributions. We aim to ***preserve correctness*** (or safety) of conservative error bounders for AVG, SUM, and COUNT aggregates ***while also providing compactness*** (for speed). We encounter a number of challenges toward this end:

1. Identifying conservative error bounder pathologies. To improve the viability of approaches with strict correctness guarantees, we must first determine the circumstances under which conservative error bounders are *too* conservative.

Our contribution: We identify two issues in range-based concentration inequalities that cause unnecessary looseness when used to compute conservative error bounds for AVG. The first, *pessimistic mass allocation* (PMA), refers to the unnecessary placement of unseen probability mass at endpoints a and b of the range enclosing the data. The second, *phantom outlier sensitivity* (PHOS), occurs when computation of the lower confidence bound g_ℓ depends on the upper range bound b *even without observed samples near* b, and vice versa for a dependency from a to g_r. PHOS captures the intuition that unobserved large (small) values should not loosen g_ℓ (g_r).

2. Correcting error bounder pathologies. After identifying correctable issues with existing conservative error bounders, we need to develop novel techniques that address these issues, while ensuring that these techniques are efficient in terms of computation and memory.

Our contribution: We develop a simple and general error bounding technique, *range trimming*, that corrects PHOS without sacrificing desirable PAC semantics. At a high level, range trimming operates by making error bounders *asymmetric*, so that g_ℓ depends only on the MAX value seen (and not on b), and g_r depends only on the MIN value seen, yielding tighter intervals when (MAX − MIN) is smaller than $(b-a)$. Range trimming can be used with any existing conservative range-based error bounder (i.e., an error bounder whose only assumption is that data falls in $[a,b]$). We show how range trimming can be used to develop an error bounder for AVG (and by extension SUM) with neither PHOS nor PMA by using it alongside a bounder based on Bernstein's inequality.

3. Minimizing sampling overhead. In order to enjoy the benefits of early termination for queries with multiple aggregates, we need to ensure that termination is not bottlenecked on any single aggregate, allowing query processing to adaptively sample from the most informative locations on physical storage while simultaneously minimizing overhead.

Our contribution: We show how to couple our approach with a sampling-optimized column store that takes without-replacement samples in a locality-aware manner, and that leverages bitmap indexes to prioritize samples that enable earlier termination for GROUP BYs. Furthermore, although existing conservative error bounders assume knowledge of the dataset size (an unreasonable assumption when a filter of unknown selectivity is applied), we show how to circumvent this limitation by computing an upper bound on this size online.

Impact. We develop error bounding techniques that more effectively leverage distributional information of the underlying data, and that therefore often lead to tighter error bounds as compared with those yielded by typical conservative error bounders. When used in conjunction with a sampling-optimized column store for in-memory analytics, we demonstrate typical speedups on the order of 10× over both exact techniques and traditional conservative approximate techniques, all without sacrificing strong correctness guarantees.

Extensibility. While our presentation focuses on confidence intervals for queries over a single table with simple AVG aggregates, we note that our techniques are more general and can be used to facilitate SUM and COUNT aggregates, queries over views formed from joins in a snowflake schema, and queries with general UDFs — we discuss these extensions in Section 4.1 and in the appendix.

Outline. The rest of this paper is organized as follows. Section 2 discusses existing conservative error bounders and their prior usage in the DBMS literature, and develops a conceptual framework for identifying issues with these error bounders. In Section 3 we develop the theory behind our RangeTrim technique, and show how to fix issues with previous error bounders in Section 2. Section 4 addresses systems issues that appear when sampling without replacement and develops FastFrame, our sampling-optimized column store, and Section 5 empirically evaluates our techniques in the context of this system. We survey additional related work in Section 6.

2. DBMS ERROR BOUND INTEGRATION

In this section, we first describe applications of confidence intervals for facilitating query processing in a database system (§2.1). Next, we survey methods for computing error bounds with guarantees applicable to DBMS aggregates (§2.2) identify their shortcomings (§2.3) and conclude with a formal problem statement (§2.4).

2.1 DBMS CI Applications

Consider the query in Figure 1. In this query, AVG aggregates are both *displayed* as output in the query results, and are also used to *filter* the set of tuples in the output. This reflects two major applications of confidence intervals in a DBMS setting: CIs that are *explicitly used* downstream, i.e., by an analyst, or CIs that are *implicitly used* by automated processes.

Explicit Use of Downstream CIs. When approximating aggregates in a DBMS, CIs can be included in the output displayed to users.

Symbols / Terms	Descriptions
$\mathcal{D}, N, S, c, m$	Dataset, num. points in dataset (i.e. $\lvert\mathcal{D}\rvert$), sample, num. points taken (for c) or desired (for m) in sample (i.e. $\lvert S\rvert$)
$g^\star, \hat{g}, g_\ell, g_r$	True aggregate, estimate, error bounds
$a, b, \sigma^2, \widehat{\sigma}^2\ \delta, \varepsilon$	Range bounds, variance, empirical variance, error probability upper bound, error
$F, \widehat{F}, L, U$	True / empirical CDF, lower and upper bounds on true CDF
Lbound, Rbound	Confidence lower (resp. upper) bounding routines parameterized on a, b, N, and other sample state(see §2.2.2).
SSI, PMA, PHOS	Sample-size-independent, pessimistic mass allocation, phantom outlier sensitivity

Table 1: Glossary of terms / notation.

For example, the AVG aggregates belonging to the groups output by the query in Figure 1 are augmented with CIs and included in the output. Such CIs help users *reason about uncertainty in approximate answers* during analysis [32, 40].

Implicit Use of Downstream CIs. Confidence intervals have been applied towards various downstream applications, for example, to enable early stopping. Example applications include high-level accuracy contracts [59, 61] (i.e., guaranteeing query results are within ε of the correct), ranking query results [45, 56], and bounding relative error [11]. In all cases, the user need not ever observe the interval: the goal is to provide *early stopping* while ensuring *correct results*. We consider these applications later in our experiments in Section 5.

Goal. In this paper, we are primarily concerned with enabling CI *compactness* (to reduce query latency) without sacrificing CI *correctness* (thereby ensuring safety), for both explicit and implicit applications of CIs. *The major goal is therefore to develop CI techniques that are as tight as possible, while always enclosing the quantity in question.* Throughout this section and Section 3, we will focus our discussion on CIs for AVG aggregates; we will cover SUM and COUNT aggregates in Section 4.

2.2 Computing CIs in a DBMS

We now describe methods for computing error bounds with accuracy guarantees in a database system, along with any assumptions required. Relevant notation is summarized in Table 1. We begin by defining error bounders, bounds, and confidence intervals.

Definition 1 [$(1-\delta)$ error bounders and bounds]**.** *A procedure P that returns error bounds $[g_\ell, g_r]$ for some aggregate $g^\star$ given a sample is a* $(1-\delta)$ error bounder *if, across all possible samples,* $\mathbb{P}\left(g^\star \notin [g_\ell, g_r]\right) < \delta$. $[g_\ell, g_r]$ *is called the* $(1-\delta)$ confidence interval *for $g^\star$, and g_ℓ and g_r are collectively referred to as* $(1-\delta)$ error *or* confidence bounds.

In contrast with asymptotic error bounders that only satisfy $\mathbb{P}\left(g^\star \notin [g_\ell, g_r]\right) \approx \delta$ for large-enough sample sizes, the $(1-\delta)$ error bounders from Definition 1 *always* satisfy $\mathbb{P}\left(g^\star \notin [g_\ell, g_r]\right) < \delta$ *for any sample size*, so we call them *sample-size-independent* (SSI).

2.2.1 Assumptions Applicable to Data in a DBMS

In the case of AVG aggregates, all error bounding procedures require some prior knowledge about the data over which they operate – otherwise, outliers can have arbitrarily strong effects on the aggregate in question. Weaker assumptions are more general, but typically yield more conservative bounds.

In this paper, we make two assumptions about the data $\mathcal{D}$ over which queries operate: first, that every datapoint $x \in \mathcal{D}$ lies in some interval $[a, b]$; second, that datapoints can be effectively sampled *without replacement* from $\mathcal{D}$. We now discuss these assumptions in the context of prior work and show that they can be implemented effectively within real systems.

Known Range Bounds. As in prior work [40], we assume that the database catalog maintains *range bounds a* and *b* for the MIN and MAX of each continuous column, inferred, for example, during data loading. (Note that we do not require $[a, b] =$ [MIN, MAX], but only that $[a, b] \supseteq$ [MIN, MAX].) These assumptions are more applicable in the context of a database as compared with stronger distributional assumptions (e.g., that the data are normal or that they obey a tighter sub-Gaussian parameter than that implied by the range bounds [70]) and can be easily maintained in the case of insertions. We refer to bounders that assume knowledge of a and b as *range-based error bounders* throughout this paper. Furthermore, we show in the appendix (§B) that it is possible to leverage the range assumption even in the case of aggregates involving arbitrary expressions over multiple columns by first solving an optimization problem for derived range bounds a' and b' that enclose the transformed data.

Sampling Without Replacement. Estimates for AVG aggregates generally converge faster for samples taken without replacement than samples taken with replacement [66, 15]. In the context of a DBMS, sampling with replacement has traditionally been considered easier than sampling without replacement, since the system does not need to "remember" the samples already taken [60, 45]. Sampling as traditionally implemented, however, also has poor locality properties, as nearly every read operation results in a cache miss. Another approach taken in prior work [64, 75, 76, 56] is to materialize samples ahead-of-time by performing a single up-front shuffle of the entire relation, so that sampling without replacement can be implemented via a scan of the data *regardless* of any applied filters or other transformations. Since this approach is valid for multiple queries executed during ad-hoc, exploratory workloads (in contrast with approaches that use workload assumptions to pre-materialize stratified samples [34, 10]), we design our system architecture around this approach, described in more detail in Section 4.

2.2.2 State for DBMS Error Bounds

OLAP queries must operate over many tuples, so it is desirable that aggregations and their error bounders maintain small of memory footprints as possible as new tuples are examined, although we will see in Section 2.2.3 that some bounders must maintain state which grows with the number of tuples examined. To better understand implementation details for error bounders within the context of a DBMS, we present error bounders in terms of the following interface:

❶ `init_state()`: Initializes state needed for error bounds.

❷ `update_state(`S, v`)`: Given the current state S and a newly-seen value v, compute new state S'.

❸ `Lbound(`S, a, b, N, δ`)`: Return a confidence lower bound for a sample whose relevant statistics are captured in state S, assuming the sample came from a finite dataset $\mathcal{D}$ of N values in $[a, b]$. The probability that the sample leads to this function returning a value greater than AVG($\mathcal{D}$) is $< \delta$.

❹ `Rbound(`S, a, b, N, δ`)`: Symmetric to `Lbound` for the confidence upper bound. Can typically be implemented in terms of `Lbound` after a suitable transformation of S.

The state S captures information such as the count of tuples examined and the current running average, as well as anything else required by `Lbound` and `Rbound`. The state initialization and update logic is analogous to state maintenance logic for aggregate functions as implemented in existing commercial database systems [3, 5, 6, 35].

Algorithm 1: Hoeffding-Serfling error bounder [66]

```
function init_state() ❶
    return {$m$: 0, $\hat{g}$: 0};
function update_state($S, v$) ❷
    $m' \leftarrow S.m + 1$;
    $\hat{g}' \leftarrow S.\hat{g} + (v - S.\hat{g})/m'$;
    return {$m$: $m'$, $\hat{g}$: $\hat{g}'$};
function Lbound($S, a, b, N, \delta$) [66] ❸
    $\varepsilon \leftarrow (b-a) \cdot \sqrt{\frac{\log(1/\delta)}{2 \cdot S.m}} \cdot (1 - \frac{S.m-1}{N})$;
    return $S.\hat{g} - \varepsilon$;
function Rbound($S, a, b, N, \delta$) ❹
    $S.\hat{g} \leftarrow (a+b) - S.\hat{g}$;
    return $(a+b) -$ Lbound($S, a, b, N, \delta$);
```

Note that both Lbound and Rbound depend on the range bounds a and b, as well as the data size N (allowing for tighter bounds when sampling without replacement).

2.2.3 Error Bounds for Finite and Bounded Data

In this section, we review some techniques for computing confidence intervals that leverage only the assumptions discussed previously: that samples are taken without-replacement from data bounded in some a priori-known range $[a, b]$. Our goal is not to be exhaustive but representative, drawing attention to previous applications in the DB literature (and lack thereof). Further details about these bounders, such as implementation pseudocode and full restatements of relevant theorems, are available in our extended technical report [55].

Hoeffding-Serfling-based Bounder. An error bounder based on the Hoeffding-Serfling inequality [66] computes CIs whose widths depend only on the range $(b-a)$ and the number of samples m, and that have size $\mathcal{O}\left((b-a)/\sqrt{m}\right)$ (if we ignore the sampling fraction term). While asymptotically optimal for worst-case data distributed with half of the points at a and the other half at b, it is needlessly wide in practice, when few points occur near a or b. An implementation of this bounder in terms of our interface from Section 2.2.2 is given in Algorithm 1. We give a statement of the Hoeffding-Serfling inequality and derive the corresponding error bounder.

Lemma 1 (Hoeffding-Serfling Inequality [66])**.** *Let $\mathcal{D} = x_1, \ldots, x_N$ be a set of N values in $[a, b]$ with average value $\mathsf{AVG}(\mathcal{D}) = \mu$. Let $X_1, \ldots, X_N$ be a sequence of random variables drawn from $\mathcal{D}$ without replacement. For every $1 \le m \le N$ and $\varepsilon > 0$,*

$$\mathbb{P}\left(\max_{1 \le k \le m} \frac{\sum_{t=1}^{k}(X_t - \mu)}{N-k} \ge \frac{m\varepsilon}{N-m}\right) \le \delta$$

where

$$\delta = \exp\left(-\frac{2m\varepsilon^2}{(1-\frac{m-1}{N})(b-a)^2}\right)$$

By focusing on $k = m$ and inverting the probability expression, we may compute a $1-\delta$ lower confidence bound as

$$\frac{1}{m}\sum_{t=1}^{m} X_t - (b-a)\sqrt{\frac{(1-\frac{m-1}{N})(\log\frac{1}{\delta})}{2m}}$$

and likewise for a upper confidence bound (replacing "–" with "+"), so that $(1-\frac{\delta}{2})$ lower and upper confidence bounds may be combined to yield a $(1-\delta)$ confidence interval (via a union bound).

Algorithm 2: Empirical Bernstein-Serfling err. bounder [15]

```
function init_state() ❶
    return {$m$: 0, $\hat{g}$: 0, $M_2$: 0};
function update_state($S, v$) ❷
    $m' \leftarrow S.m + 1$;
    $\hat{g}' \leftarrow S.\hat{g} + (v - S.\hat{g})/m'$;
    $M_2' \leftarrow S.M_2 + v^2$;
    return {$m$: $m'$, $\hat{g}$: $\hat{g}'$, $M_2$: $M_2'$};
function Lbound($S, a, b, N, \delta$) [15] ❸
    $\kappa \leftarrow 7/3 + 3/\sqrt{2}$;
    $\rho \leftarrow \mathbb{I}\{S.m \le N/2\} \cdot (1 - \frac{S.m-1}{N})$;
    $\rho \leftarrow \rho + \mathbb{I}\{S.m > N/2\} \cdot ((1 - \frac{S.m}{N}) \cdot (1 + \frac{1}{S.m}))$;
    $\varepsilon \leftarrow \sqrt{S.M_2/S.m - S.\hat{g}^2} \cdot \sqrt{\frac{2\rho \cdot \log(5/\delta)}{S.m}} + \kappa \cdot (b-a) \cdot \frac{\log(5/\delta)}{S.m}$;
    return $S.\hat{g} - \varepsilon$;
function Rbound($S, a, b, N, \delta$) ❹
    $S.\hat{g} \leftarrow (a+b) - S.\hat{g}$;
    return $(a+b) -$ Lbound($S, a, b, N, \delta$);
```

Empirical Bernstein-Serfling-based Bounder. A concentration inequality for sampling without replacement given in [15], the *Bernstein-Serfling* inequality assumes knowledge of both $(b-a)$ and $\mathsf{VAR}(\mathcal{D}) = \sigma^2 = \frac{1}{N}\sum_{x \in \mathcal{D}}(x - \mathsf{AVG}(\mathcal{D}))^2$. We defer a statement of the full result to the appendix. Here we note that inverting the inequality gives error bounds as

$$\frac{1}{m}\sum_{t=1}^{m} X_t \pm \mathcal{O}\left(\sigma/\sqrt{m} + (b-a)/m\right)$$

if we again ignore the sampling fraction term. Comparing these error bounds to those of Hoeffding-Serfling, which has widths of size $\mathcal{O}\left((b-a)/\sqrt{m}\right)$ (again ignoring the sampling fraction), we see that error bounds derived from the Bernstein-Serfling inequality can be significantly tighter when σ is small compared to $(b-a)$.

Knowledge of $\mathsf{VAR}(\mathcal{D})$ typically cannot be assumed in a setting where $\mathsf{AVG}(\mathcal{D})$ is unknown. Fortunately, there also exists an *empirical* variant of the Bernstein-Serfling inequality (also given in [15], like the non-empirical variant). The analysis for the empirical Bernstein-Serfling inequality proceeds by augmenting the analysis for the non-empirical variant with a concentration inequality relating the estimator $\widehat{\sigma}^2 = \frac{1}{m}\sum_{t=1}^{m}(X_t - \bar{X})^2$ to $\mathsf{VAR}(\mathcal{D})$. We again deferring the full statement to the appendix. This yields $(1-\delta)$ error bounds given by

$$\frac{1}{m}\sum_{t=1}^{m} X_t \pm \mathcal{O}\left(\widehat{\sigma}/\sqrt{m} + (b-a)/m\right)$$

Note that these error bounds differ from the those of the non-empirical variant only in that σ is replaced by $\widehat{\sigma}$ (modulo slightly worse constants hidden by the asymptotic notation). Although $\widehat{\sigma}$ is a random quantity, it concentrates near σ, so that an error bounder based on the empirical Bernstein-Serfling bound returns bounds of asymptotically the same width as those returned by an error bounder based on the non-empirical variant and with full access to σ^2, w.h.p. Algorithm 2 gives an implementation of an empirical Bernstein-Serfling-based error bounder in terms of our interface from Section 2.2.2. Note that Algorithm 2 as presented shows computation of the sample variance in terms of the second moment $M_2 = \sum v^2$ for the sake of exposition; a real implementation might use a more numerically stable one-pass algorithm for the variance [73, 20, 51].

Anderson/DKW-based Bounder. Anderson described a way to compute distribution-free / nonparametric error bounds for the mean given error bounds for the cumulative distribution function (CDF) in [13]. Denoting the true and empirical CDF for some distribution supported on $[a, b]$ with F and $\widehat{F}$, respectively, Anderson showed

Algorithm 3: Anderson/DKW error bounder [27, 13, 57]

```
function init_state() ❶
    return {}

function update_state($S, v$) ❷
    return $S \cup \{v\}$

function Lbound($S, a, b, N, \delta$) ❸
    $\varepsilon \leftarrow \sqrt{\frac{\log(1/\delta)}{2\cdot|S|}}$;
    $\widehat{F} \leftarrow$ empirical CDF based on $S$;
    $S' \leftarrow \{x \in S : \widehat{F}(x) \leq 1 - \varepsilon\}$;
    return $\varepsilon \cdot a + (1 - \varepsilon) \cdot \mathsf{AVG}(S')$;

function Rbound($S, a, b, N, \delta$) ❹
    return $(a + b) -$ Lbound($(a + b) - S, a, b, N, \delta$);
```

how to use high-probability bounds α and β such that

$$\widehat{F} - \alpha \preceq F \preceq \widehat{F} + \beta$$

to get high-probability bounds on the mean of F. To see how, recall the following identity:

Lemma 2. *Consider a CDF F supported on $[a, b]$. Then the mean μ of the distribution corresponding to F satisfies*

$$\mu = b - \int_a^b F(x)dx$$

Thus, given lower and upper bounds L and U on the CDF F that satisfy $\forall x \in [a, b], L(x) \leq F(x) \leq U(x)$, error bounds around the mean may be computed as

$$\left[b - \int_a^b U(x)dx \ , \quad b - \int_a^b L(x)dx \right]$$

since $L \preceq F \preceq U$ implies $-U \preceq -F \preceq -L$.

Anderson used the Dvoretzky-Kiefer-Wolfowitz (DKW) inequality [27] to compute α and β. Informally, DKW states that the empirical CDF $\widehat{F}$ computed from i.i.d. samples taken from a distribution with CDF F concentrates around F everywhere:

Lemma 3 (DKW Inequality [27, 57])**.** *Let $X_1, \ldots, X_m \overset{iid}{\sim} F$, and let $\widehat{F}$ be the empirical CDF corresponding to the sample $\{X_i\}$. Then for every $\varepsilon > 0$,*

$$\mathbb{P}\left(\sup_{t \in dom(F)} |\widehat{F}(t) - F(t)| > \varepsilon \right) \leq 2 \exp\left(-2m\varepsilon^2\right)$$

The DKW inequality provides a method to obtain the values of α and β, since it implies that

$$\widehat{F} - \sqrt{\frac{\log 2/\delta}{2m}} \preceq F \preceq \widehat{F} + \sqrt{\frac{\log 2/\delta}{2m}}$$

with probability greater than $1 - \delta$. At the time [13] was published, however, the constant in front of the DKW inequality had not yet been proved by Massart [57], so it appears that Anderson computed α and β using a lookup table.

Although Lemma 3 as stated applies for sampling with replacement from an infinite population, please see Appendix C for a proof that DKW still holds when $X_1, \ldots, X_m$ are drawn without replacement from a finite population of size N, for any $N > 0$, stated as the following theorem:

Theorem 1. *For any $N > 0$, the DKW inequality applies for sampling without replacement from a finite dataset of size N.*

Error Bounder	PMA	PHOS	Sampling	Memory
Hoeffding(-Serfling)	✓	✓	R* (NR)	$\mathcal{O}(1)$
Berstein(-Serfling)	🟩	✓	R* (NR)	$\mathcal{O}(1)$
Anderson/DKW [55]	✓	🟩	R, NR	$\mathcal{O}(m)$

Table 2: Summary of properties exhibited by various error bounders. R = sampling with replacement, NR = without. A * indicates that the non-Serfling variant also holds for NR sampling.

The procedure just described for computing error bounds around the mean of a distribution given i.i.d. samples thus also works for computing error bounds around $\mathsf{AVG}(\mathcal{D})$ given without-replacement samples from the finite dataset $\mathcal{D}$. It is presented in terms of our interface from Section 2.2.2 in Algorithm 3.

Applications in Prior DB Literature. To our knowledge, Hoeffding and Hoeffding-Serfling-based bounders are the only SSI bounders that have seen extensive use in the DB literature for computing error bounds for AVG [45, 11, 40, 36]. We are aware of one incorrect application of the empirical Bernstein-Serfling inequality [24] (incorrect because the procedure given in [24] continuously recomputes confidence $(1 - \delta)$ intervals as more samples are taken, so that the overall procedure is no longer guaranteed to fail with probability at most δ). Overall it is somewhat surprising that error bounders derived from the empirical Bernstein-Serfling inequality [15] have not seen more widespread usage, as they are nearly as simple to compute as those derived from the Hoeffding-Serfling inequality and typically yield error bounds that are much tighter.

2.3 Error Bounder Pathologies

We identify two problems that cause SSI error bounders to be *too* conservative. These pathologies, which we refer to as *pessimistic mass allocation (PMA)* and *phantom outlier sensitivity (PHOS)*, are based on simple intuitions about how error bounders should behave: namely, they should return tighter bounds when observing samples with fewer extreme values, and error lower bounds (respectively error upper bounds) should only be looser due to potential large values (resp. small values) if such values are actually observed.

2.3.1 Pessimistic Mass Allocation

PMA captures the intuition that error bounders should be sensitive to the observed sample values:

Definition 2 [PMA]**.** *An error bounding procedure P exhibits* pessimistic mass allocation (PMA) *if there exists a dataset $\mathcal{D}$ bounded in $[a, b]$, a value a' with $a < a' < b$, and a set $S \subseteq \mathcal{D}$ with values in $[a, a')$ such that, for $S' = \{\max(x, a') : x \in S\}$, P returns a confidence interval of the same width for both S and S'. P likewise exhibits PMA if there exists some b' with $a < b' < b$ and an S with values in $(b', b]$ such that, for $S' = \{\min(x, b') : x \in S\}$, P returns a confidence interval of the same width for both S and S'.*

That is, for an error bounder P with PMA, we can replace the smallest (largest) elements in a sample with something larger (resp. smaller) without shrinking the width of P's returned confidence interval. For example, for data known to lie in $[0, 1]$, P might yield an interval of the same width for both a sample split evenly between 0 and 1 as well as a sample split evenly between 0.25 and 0.75, even though the latter sample should clearly give rise to a tighter interval. Intuitively, P is overly-pessimistic about how mass in the underlying distribution from which it is sampling is allocated, despite contrary evidence observed in the sample.

2.3.2 Phantom Outlier Sensitivity

PHOS captures the intuition that unobserved extreme values should not affect both the lower and the upper error bounds computed by some error bounder P:

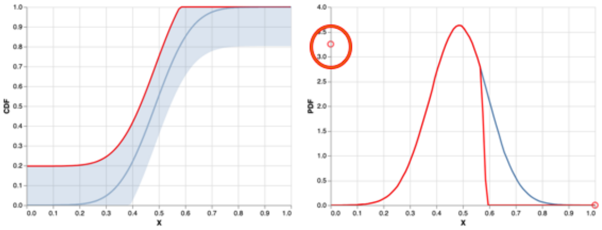


Figure 3: Error bounds from the DKW inequality exhibit pessimistic mass allocation.

Definition 3 [PHOS]**.** *An error bounding procedure P exhibits* phantom outlier sensitivity (PHOS) *if, for data falling in $[a, b]$, P's returned confidence lower bound g_ℓ depends on the value of b, and similarly if the g_r returned by P depends on a.*

To understand PHOS intuitively, consider the case of computing a confidence lower bound. Given a sample S, the worse P "believes" S could be shifted (on average) toward larger values as compared to $\mathcal{D}$, the smaller of a confidence lower bound it should return. In what ways could S be shifted toward higher values? One possibility is if small elements are underrepresented in S. The other possibility, and the one we are interested in, is if large elements are overrepresented in S. For this reason, a confidence lower bound should only be affected by datapoints near the upper range bound b *if it actually observes them*, and the appearance of b in the computation of a confidence lower bound is a potential source of unnecessary conservativeness.

2.3.3 Examples of PMA *and* PHOS *in Error Bounders*

In this section, we give examples of PMA and PHOS in the context of previously-discussed error bounders. Table 2 summarizes pathologies exhibited by various SSI error bounders.

Hoeffding-based. Hoeffding-based error bounders suffer from both PMA and PHOS. They have PMA since their returned CIs have widths depending only on the range of the data, $(b - a)$, and the number of samples. As such, replacing values in the sample with larger or smaller values does not affect the width of the returned error bounds. Such bounders also have PHOS since they have symmetric error, with both ends of the confidence interval depending on both range bounds a and b.

Berstein-based. Bernstein-based error bounders do *not* suffer from PMA. To see this, notice that increasing the smallest values in some sample will also reduce the sample variance, affecting the width of the returned confidence interval, and likewise for decreasing the largest values in the sample. These bounders do, however, suffer from PHOS. Like Hoeffding-based bounders, they return confidence intervals with symmetric error, so that each end of the confidence interval is affected by both ends of the data range a and b.

Anderson/DKW-based. Anderson/DKW-based error bounders are interesting in that they suffer from PMA, but not PHOS. Consider the ε mass unaccounted for when computing a confidence lower bound using an Anderson/DKW-based bounder. As shown in Figure 3, it all goes toward to lower range bound, a, which is sufficient for PMA. On the other hand, where does it come from? It comes from the ε-fraction largest observed points. This does not depend at all on the value of the upper range bound b, indicating that the confidence lower bound does not suffer from PHOS. Symmetric statements hold for the confidence upper bound, of course.

2.4 Problem Statement

We are now ready to give a formal problem statement.

Problem 1. *Design an SSI error bounder that, given a without-replacement sample from any $\mathcal{D}$ with elements from $[a, b] \subseteq \mathbb{R}$, suffers from neither PMA nor PHOS when computing $(1 - \delta)$ error bounds for AVG$(\mathcal{D})$, for any $0 < \delta < 1$.*

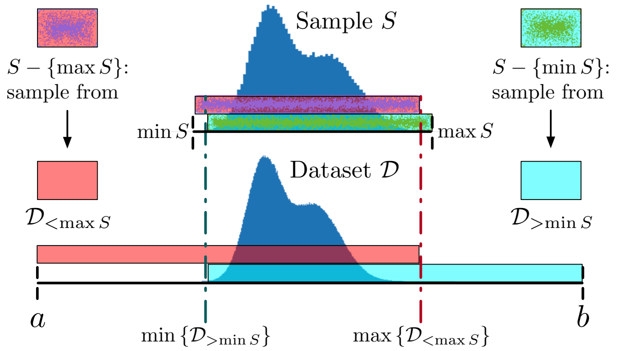


Figure 4: Range trimming eliminates PHOS for range-based error bounders.

Our solution to Problem 1 is given in Section 3 and relies on a technique we call *range trimming* in order to systematically eliminate PHOS from any range-based error bounder.

Although the solution as presented in Section 3 additionally assumes knowledge of the size of $\mathcal{D}$, Section 4 shows how our real-world implementation circumvents this limitation.

3. FIXING BOUNDER PATHOLOGIES

From our discussion in Section 2.3, we see that there do exist error bounders without one of either PMA or PHOS, but not without both. We first argue that error bounders without PHOS must be *asymmetric*; that is, they cannot compute bounds of the form $\hat{g} \pm \varepsilon$, where the same ε is both added and subtracted to the sample average $\hat{g}$ in order to compute bounds. Next, we describe how to use a process we call *range trimming* to convert any symmetric, ranged-based error bounder to an asymmetric one without PHOS.

3.1 Decoupling Lower and Upper Bounds

Excepting an error bounder based on DKW, all of the error bounders surveyed suffer from PHOS. This is because all the other error bounders are based on concentration inequalities with *symmetric error* — that is, they return confidence intervals $[g_\ell, g_r]$ of the form $[\hat{g} - \varepsilon, \hat{g} + \varepsilon]$. At a high level, it is precisely this symmetry that causes PHOS. Although a confidence lower bound should not have any dependency on b, it is intuitively unavoidable that it has some dependency on a. Reiterating, an estimate $\hat{g}$ could be an overestimate because of (i) not enough observed values near a, or (ii) too many observed values near b. A similar statement holds regarding confidence upper bounds, with the roles of a and b reversed.

We hypothesize that it is impossible for any confidence lower bound (resp. upper bound) to completely eliminate the dependency on a (resp. b), since it is always possible that the confidence bounding procedure got "unlucky" and operated on a sample in which values near a (resp. b) were underrepresented. Taking this hypothesis as given, this means that any symmetric confidence bounding procedure that returns bounds of the form $[\hat{g} - \varepsilon, \hat{g} + \varepsilon]$ will have ε dependent on both a and b — that is, any symmetric confidence bounding procedure will have PHOS. As such, the first step to eliminating PHOS from range-based confidence bounders is to accept asymmetric error as a hard requirement: that is, we must consider confidence bounding procedures that return bounds of the form $[\hat{g} - \varepsilon_\ell, \hat{g} + \varepsilon_r]$ for which ε_ℓ and ε_r are not necessarily equal.

3.2 Range Trimming

Our approach to deriving an error bounder with neither PMA nor PHOS is to start with a symmetric bounder without PMA (such as that of Algorithm 2) and "asymmetrize" it so that Lbound becomes

Algorithm 4: The RangeTrim meta-algorithm

```
Input  : Dataset $\mathcal{D}$ of $N$ values in $[a, b]$, error prob. $\delta$, sample size $m$
Output : Error bounds that fail to enclose $\mathsf{AVG}(\mathcal{D})$ with probability $< \delta$

$S_\ell \leftarrow$ init_state();
$S_r \leftarrow$ init_state();
$a' \leftarrow$ sample_without_replacement($\mathcal{D}$);
$b' \leftarrow a'$;
for $i = 1$ to $m - 1$ do
    $v \leftarrow$ sample_without_replacement($\mathcal{D}$);
    $S_\ell \leftarrow$ update_state($S_\ell$, $\min(v, b')$);
    $S_r \leftarrow$ update_state($S_r$, $\max(v, a')$);
    $a' \leftarrow \min(a', v)$;
    $b' \leftarrow \max(b', v)$;
end
return [Lbound($S_\ell, a, b', N - 1, \frac{\delta}{2}$), Rbound($S_r, a', b, N - 1, \frac{\delta}{2}$)];
```

independent of b, and Rbound becomes independent of a, thereby eliminating PHOS. The result, given in Algorithm 4, wraps any existing range-based error bounder.

Besides the memory required to maintain state for the left and right error bounders, S_ℓ and S_r, Algorithm 4 requires $\mathcal{O}(1)$ extra memory to maintain the MIN and MAX element seen so far (which replace a and b when computing Rbound and Lbound, respectively).

When $\mathcal{D}$ contains unique elements, Algorithm 4 conceptually performs the following steps:

1. Sample S without replacement from $\mathcal{D}$.
2. Use Lbound to compute a $1 - \frac{\delta}{2}$ lower confidence bound for $\mathsf{AVG}(\mathcal{D}_{<\max S})$, with $S - \{\max S\}$ as the sample, and with a and $\max S$ in place of the normal range bounds a and b, respectively.
3. Use Rbound to compute a $1 - \frac{\delta}{2}$ upper confidence bound for $\mathsf{AVG}(\mathcal{D}_{>\min S})$, with $S - \{\min S\}$ as the sample, and with $\min S$ substituted for the range bound lower bound a.

Note that we use $\mathcal{D}_{<x}$ and $\mathcal{D}_{>x}$ as shorthand for $\mathcal{D} \cap (-\infty, x)$ and $\mathcal{D} \cap (x, \infty)$, respectively. The primary difference between these high-level steps and the pseudocode presented in Algorithm 4 is that Algorithm 4 maintains $\min S$ and $\max S$ in an online, streaming fashion (so that the sample S does not need to be stored in memory), and that the confidence interval returned by Algorithm 4 is valid even when $\mathcal{D}$ contains duplicates (although the returned confidence bounds will bound the AVG of sets that differ slightly from $\mathcal{D}_{<\max S}$ and $\mathcal{D}_{>\min S}$). That said, we restrict our discussion and analysis to the case where $\mathcal{D}$ contains unique elements, for simplicity.

Correctness of Algorithm 4 crucially depends on the fact that, conditioned on the value of $\max S$ (and for any such value), the remaining elements in S (namely $S - \{\max S\}$) constitute a uniform without-replacement sample from $\mathcal{D}_{<\max S}$, with a symmetric statement for $\min S$ and $S - \{\min S\}$. At a high level, this means that a confidence lower bound computed over $S - \{\max S\}$ is a valid confidence lower bound for $\mathsf{AVG}(\mathcal{D}_{<\max S})$, and since $\mathsf{AVG}(\mathcal{D}_{<\max S}) \le \mathsf{AVG}(\mathcal{D})$, it is also a valid confidence lower bound for $\mathsf{AVG}(\mathcal{D})$, with symmetric statements holding for the confidence upper bound, $S - \{\min S\}$, and $\mathcal{D}_{>\min S}$. These core ideas are illustrated in Figure 4.

Tradeoffs. When either of the range bounds a or b are actually observed in a sample, Algorithm 4 will compute CIs that are looser than necessary. For example, suppose that, for $[a, b] = [0, 1]$, a sample with a single 0 is observed. When computing the upper confidence bound, Algorithm 4 sets $a' = 0$, which is the same as the original a; however, it also throws away this 0 point which would have pulled the upper confidence bound lower. The upper confidence bound actually computed will end up being larger than necessary, corresponding to an unnecessarily loose interval. Fortunately, we have found that, in practice, losing this single sample does not significantly increase the number of samples needed to achieve some desired CI width.

3.3 Proof of Correctness

In this section, we prove correctness of Algorithm 4 (that is, that it returns intervals that fail to enclose $\mathsf{AVG}(\mathcal{D})$ with probability less than δ). For the sake of simplicity, our analysis assumes that $\mathcal{D}$ contains no duplicate values, although we show how to remove this assumption at the end of this section. To begin, we first prove a crucial lemma about the sampling distribution of $S - \{\max S\}$, given that S was sampled uniformly without-replacement from $\mathcal{D}$.

Lemma 4. *Given a dataset $\mathcal{D}$ of N unique real values in $[a, b]$ and a uniform without-replacement sample S of m values from $\mathcal{D}$, if we denote $b' = \max S$, the set $S - \{b'\}$ takes the distribution of a uniform without-replacement sample from $\mathcal{D}_{<b'} = \mathcal{D} \cap [a, b')$, for any applicable value of $b' \in \mathcal{D}$.*

Proof. Because S is drawn uniformly without-replacement from $\mathcal{D}$, any particular instance satisfies

$$\mathbb{P}_{\mathcal{D}}[S = s] = \binom{|\mathcal{D}|}{|s|}^{-1} \mathbb{I}\{s \subseteq \mathcal{D}\} = \binom{N}{m}^{-1} \mathbb{I}\{s \subseteq \mathcal{D}\}$$

where we use the notation $\mathbb{P}_{\mathcal{D}}[S = s]$ to denote the probability that s was drawn uniformly without-replacement from $\mathcal{D}$, and $\mathbb{I}\{\cdot\}$ denotes the indicator function. We need to show that, for any $b' \in \mathcal{D}$,

$$\mathbb{P}_{\mathcal{D}}\left[S = s \,|\, \max S = b'\right] = \mathbb{P}_{\mathcal{D}_{<b'}}\left[S = s - \{b'\}\right] \mathbb{I}\left\{\max(s) = b'\right\}$$

First, letting s' be any set such that $|s'| = m - 1$, we have that

$$\mathbb{P}_{\mathcal{D}_{<b'}}\left[S = s'\right] = \binom{|\mathcal{D}_{<b'}|}{m-1}^{-1} \mathbb{I}\left\{s' \subseteq \mathcal{D}_{<b'}\right\}$$

Next, consider $\mathbb{P}_{\mathcal{D}}\left[S = s \,|\, \max S = b'\right]$. Bayes' rule gives that

$$\mathbb{P}_{\mathcal{D}}\left[S = s \,|\, \max S = b'\right] = \frac{\mathbb{P}_{\mathcal{D}}\left[S = s \wedge \max S = b'\right]}{\mathbb{P}_{\mathcal{D}}[\max S = b']}$$

We have $\mathbb{P}_{\mathcal{D}}\left[S = s \wedge \max S = b'\right] = \mathbb{P}_{\mathcal{D}}[S = s]\, \mathbb{I}\left\{\max(s) = b'\right\}$ which is a known quantity, so the key is to compute the denominator $\mathbb{P}_{\mathcal{D}}\left[\max S = b'\right]$. Using the assumption that $\mathcal{D}$ contains unique elements, we may proceed by analogy with binary strings. The rank of b' within $\mathcal{D}$ (starting from the smallest element) is $1 + |\mathcal{D}_{<b'}|$, so we need to compute the number of binary strings of length N containing m 1's and $(N - m)$ 0's such that position $1 + |\mathcal{D}_{<b'}|$ has a 1, and the remaining $(m - 1)$ 1's are all at positions less than $1 + |\mathcal{D}_{<b'}|$. This is precisely the same as the number of binary strings of length $|\mathcal{D}_{<b'}|$ with $(m - 1)$ 1's and $(|\mathcal{D}_{<b'}| - m + 1)$ 0's. Putting everything together,

$$\begin{aligned}
\mathbb{P}_{\mathcal{D}}\left[S = s \,|\, \max S = b'\right] &= \frac{\mathbb{P}_{\mathcal{D}}[S = s]\, \mathbb{I}\left\{\max(s) = b'\right\}}{\mathbb{P}_{\mathcal{D}}[\max S = b']} \\
&= \frac{\binom{N}{m}^{-1} \mathbb{I}\left\{s \subseteq \mathcal{D} \wedge \max(s) = b'\right\}}{\binom{|\mathcal{D}_{<b'}|}{m-1} / \binom{N}{m}} \\
&= \binom{|\mathcal{D}_{<b'}|}{m-1}^{-1} \mathbb{I}\left\{s \subseteq \mathcal{D} \wedge \max(s) = b'\right\} \\
&= \binom{|\mathcal{D}_{<b'}|}{m-1}^{-1} \mathbb{I}\left\{s - \{b'\} \subseteq \mathcal{D}_{<b'} \wedge \max(s) = b'\right\} \\
&= \mathbb{P}_{\mathcal{D}_{<b'}}\left[S = s - \{b'\}\right] \mathbb{I}\left\{\max(s) = b'\right\}
\end{aligned}$$

which is precisely what we wanted to show. □

Wrinkle in Lemma 4 and Fix. The proof of Lemma 4 assumes unique values; we show here how to remove this assumption without loss of generality. The uniqueness assumption as used is necessary only to ensure that elements of $\mathcal{D}$ are *totally ordered* under some relation "$\prec$" (with "$\prec$" ≡ "$<$" in the proof). To fix, we can simply augment every $v \in \mathcal{D}$ with an additional unique *label* (where the set of labels are totally ordered) such that item v becomes $v' \equiv (v, v_i)$. Then, define "$\prec$" as a relation such that $v' \prec w'$ if $v < w$, or $v = w$ and $v_i < w_i$. In this way, any $v', w' \in \mathcal{D}'$ satisfy exactly one of $v' \prec w'$ or $w' \prec v'$, and the proof of Lemma 4 goes through, replacing $\mathcal{D}$ with $\mathcal{D}'$ and "$<$" with "$\prec$" where appropriate.

We next give a symmetric statement for $S - \{\min S\}$ and $\mathcal{D}_{>\min S}$ as the below corollary:

Corollary 1. *Given a dataset $\mathcal{D}$ of N unique real values in $[a, b]$ and a uniform without-replacement sample S of m values from $\mathcal{D}$ such that $\min S = a'$, the set $S - \{a'\}$ is a uniform without-replacement sample from $\mathcal{D}_{>a'} = \mathcal{D} \cap (a', b]$, for any applicable value of $a' \in \mathcal{D}$.*

Monotonicity Property and Correctness Proof. Before proving the main result, we briefly describe the *dataset size monotonicity property* obeyed by all bounders in this paper. This fact will be used in the main correctness proof. When N is unknown, an upper bound on N suffices, because bounders in this paper all satisfy the following: for any S, a, b, N, δ, and $N' > N$,

$$\texttt{Lbound}(S, a, b, N', \delta) \le \texttt{Lbound}(S, a, b, N, \delta)$$

$$\texttt{Rbound}(S, a, b, N', \delta) \ge \texttt{Rbound}(S, a, b, N, \delta)$$

That is, using an upper bound for N can only make the CI looser, and since SSI range-based error bounders with the correct dataset size N fail with probability at most δ, they must also fail with probability at most δ for any $N' > N$.

We are now ready to prove correctness of Algorithm 4.

Theorem 2. *Given SSI range-based bounders* Lbound *and* Rbound *for computing lower (resp. upper) confidence bounds and a dataset $\mathcal{D}$ of N unique values known to all fall in the interval $[a, b] \subseteq \mathbb{R}$, Algorithm 4 returns a $(1 - \delta)$ confidence interval for* AVG$(\mathcal{D})$.

Proof. Algorithm 4 proceeds by drawing S uniformly and without replacement from $\mathcal{D}$ and computing $a' = \min S$, $b' = \max S$, S_ℓ, and S_r, where the latter two quantities capture relevant statistics from the sample $S - \{b'\}$ and $S - \{a'\}$, respectively, so we treat S_ℓ and S_r as if $S_\ell = S - \{b'\}$ and $S_r = S - \{a'\}$. By Lemma 4, we have that S_ℓ is a uniform sample of $m - 1$ values drawn without replacement from $\mathcal{D}_{<b'}$, and likewise by Corollary 1 S_r is a uniform sample of $m - 1$ values drawn without replacement from $\mathcal{D}_{>a'}$. Because Lbound and Rbound are assumed to be SSI, range-based error bounders, we have that

$$\mathbb{P}\left(\texttt{Lbound}(S_\ell, a, b', N-1, \frac{\delta}{2}) > \text{AVG}(\mathcal{D})\right) \tag{1}$$

$$\le \mathbb{P}\left(\texttt{Lbound}(S_\ell, a, b', |\mathcal{D}_{<b'}|, \frac{\delta}{2}) > \text{AVG}(\mathcal{D})\right) \tag{2}$$

$$\le \mathbb{P}\left(\texttt{Lbound}(S_\ell, a, b', |\mathcal{D}_{<b'}|, \frac{\delta}{2}) > \text{AVG}(\mathcal{D}_{<b'})\right) < \frac{\delta}{2} \tag{3}$$

and symmetrically for $\texttt{Rbound}(S_r, a', b, N - 1, \frac{\delta}{2})$, but with "$>$" replaced with "$<$" in the probability expression above, and replacing $\mathcal{D}_{<b'}$ with $\mathcal{D}_{>a'}$. (1) → (2) follows from the dataset size monotonicity property of Lbound (§2.2.2), applicable since $N - 1 \ge |\mathcal{D}_{<b'}|$, and (2) → (3) follows since $\text{AVG}(\mathcal{D}_{<b'}) \le \text{AVG}(\mathcal{D})$, as the former is clipped above b' (and similarly for Rbound since $\text{AVG}(\mathcal{D}_{>a'}) \ge \text{AVG}(\mathcal{D})$). Union bounding over the cases for each of Lbound and Rbound, the probability that Algorithm 4 returns an interval that does not enclose AVG$(\mathcal{D})$ is at most δ. □

4. SYSTEM CONSIDERATIONS

In this section, we address a number of implementation issues that become pertinent when applying techniques of previous sections in a real system. Although the techniques presented in this section are auxiliary to our primary contribution and can be used with any CI approach, they are developed with SSI error bounders and strong probabilistic guarantees in mind. First, we describe how to augment the techniques of Section 3, which apply for a fixed sample size taken without replacement from a finite dataset of known size, with locality-aware *scan-based* without-replacement sampling that need not know N, and we further describe how to use this layout to facilitate SUM and COUNT aggregations (§4.1). Next, we describe an *optional stopping* routine that does not require a sample size to be specified up-front (§4.2). Finally, we describe an *active scanning* architectural optimization that prioritizes samples that facilitate early termination (§4.3), all without losing guarantees proved in Section 3.

These system details are implemented within the context of FastFrame, which is our general relational column store for approximate report generation with guarantees. FastFrame uses the error bounders from Section 3 and pairs them with a practical architecture for without-replacement sampling. FastFrame uses block-based bitmaps over categorical attributes (similar to [56]) for efficient processing of queries with predicates or groups. Furthermore, for continuous attributes, FastFrame stores the minimum and maximum values in a catalog, to be used as the range bounds a and b for the desired range-based error bounder.

4.1 Scan-Based Sampling for DB Aggregates

We now describe how FastFrame implements without-replacement sampling in a locality-aware manner by *scanning* over pre-shuffled data, and furthermore how this approach can be used to compute CIs for COUNT and SUM. The up-front shuffling cost need only be paid once in order to facilitate many queries, although care must be taken to set the error probability δ small enough when running multiple queries to avoid losing error bounder guarantees. This approach is not new and has been used in prior work [56, 64, 75, 76, 31]. We begin by introducing *scrambles* and *aggregate views*:

Definition 4 [Scramble]**.** *A* scramble *is an* ordered *copy of a relational table that has been permuted randomly, allowing for scan-based without-replacement sampling.*

Note that there exist external shuffling techniques that can handle data too large to fit in memory [48].

Scanning a continuous column in a scramble is equivalent to sampling without replacement. In fact, scanning any subset of data in a continuous column in a scramble (assuming the subset is chosen without knowledge of the order of data) is also equivalent to sampling without replacement, so that scanning a scramble can be used to sample without replacement for any aggregate appearing in a query containing arbitrary filters or GROUP BY clauses. We call such subsets *aggregate views*:

Definition 5 [Aggregate View]**.** *An* aggregate view *for some aggregate A appearing in a query (possibly belonging to a group induced by a GROUP BY clause) is the set of values in a scramble that contribute toward the computation of A.*

Choosing δ to Maintain Guarantees. Note that δ must be divided by the number of aggregate views in a query (or an upper bound) to preserve error guarantees (via a union bound). Furthermore, to guarantee (via another union bound) that the probability of one or more queries in a workload giving incorrect results is at most δ, we must further divide by (an upper bound on) the number of queries in the workload before supplying it as a parameter to an error bounder. For this reason, we choose $\delta = 10^{-15}$ when we discuss our empirical study in Section 5, since even for large workloads comprised of, say, 10^7 queries, the probability of one or more mistakes will still be at most 10^{-8}. Indeed, our techniques are specifically targeted toward the case wherein it is known ahead of time that a large (but possibly unknown) number of exploratory queries will be issued, so that the cost to materialize the scramble is justified. Note that, provided δ is properly decayed, guarantees hold when the same scramble is used across an entire query workload; i.e., it is not necessary to reshuffle the scramble between queries.

Computing CIs for COUNT. Ensuring that data in a scramble are permuted randomly makes it easy to compute bounds on the selectivities of aggregate views, and by extension on the COUNT of tuples in each aggregate view, using existing techniques [37, 38]. To outline the basic idea, consider that one can conceptually assign each row of a scramble a 1 if it belongs to the aggregate view of interest, and a 0 otherwise. The AVG of this "derived" view (over the whole scramble) is exactly the selectivity of the aggregate view, and we can use a Hoeffding-Serfling-based bounder to compute a CI for the selectivity (using range bounds of $a = 0$ and $b = 1$). Multiplying these bounds by the total number of rows in the scramble then yields a CI for the COUNT of rows that participate in the aggregate view.

In more detail, for a scramble with R rows, N of which are in the sample view V for a query Q, the number of rows seen that belong to V, m_v, after scanning r rows of the scramble is a hypergeometric random variable [18] whose mean is the selectivity σ_v of V multiplied by r, $\sigma_v \cdot r$. One could use bounds specifically tailored to the hypergeometric distribution (or even perform an exact computation) to compute an upper bound on σ_v that holds w.h.p., but in this work we use a simple strategy that uses Hoeffding-Serfling to bound σ_v, stated as follows.

Lemma 5. *The probability that a scan of a scramble of size R that has processed r rows so far sees fewer than $(\sigma_v - \varepsilon) \cdot r$ or more than $(\sigma_v + \varepsilon) \cdot r$ rows belonging to V is at most δ, for $\varepsilon = \sqrt{\frac{\log(2/\delta)}{2r} \cdot (1 - \frac{r-1}{R})}$.*

Proof. Follows immediately from application of the Hoeffding-Serfling inequality [66]. □

Lemma 5 implies that, for a scan that has seen m_v rows so far belonging to V, σ_v is within

$$\widehat{\sigma}_v \pm \varepsilon = \frac{m_v}{r} \pm \sqrt{\frac{\log(2/\delta)}{2r} \cdot \left(1 - \frac{r-1}{R}\right)}$$

w.h.p. This in turn implies that N, the number of tuples belonging to V, is within $[N^-, N^+] = [(\widehat{\sigma}_v - \varepsilon) \cdot R, (\widehat{\sigma}_v + \varepsilon) \cdot R]$, w.h.p.

Combining Lemma 5 with error bounders to compute CIs for AVG with unknown COUNT. For error bounders Lbound and Rbound of the form described in Section 3 that require the data range bounds a and b as well as the data size N, a $(1 - \delta)$ confidence interval is computed as

$$[\texttt{Lbound}(S, a, b, N, \delta/2),\ \texttt{Rbound}(S, a, b, N, \delta/2)]$$

The following theorem describes how to use Lemma 5 to compute $(1 - \delta)$ error bounds when N is unknown.

Theorem 3. *Consider a query Q operating over some size-R scramble with corresponding sample view V. Suppose a scan the scramble that has processed k rows so far has seen m_v rows belonging to V (from which S is computed). Letting*

$$N^+ = \left(\frac{m_v}{r} + \sqrt{\frac{\log(1/(1-\alpha)\cdot\delta)}{2r} \cdot \left(1 - \frac{r-1}{R}\right)}\right) \cdot R$$

then the interval

$$\left[\texttt{Lbound}(S, a, b, N^+, \alpha \cdot \delta/2),\ \texttt{Rbound}(S, a, b, N^+, \alpha \cdot \delta/2)\right]$$

is a $(1 - \delta)$ confidence interval for the mean of V, for any $\alpha \in (0, 1)$.

Proof. Conditioning over whether $N \le N^+$ or $N > N^+$, the probability that the aforementioned interval $[L, R]$ fails to contain the desired mean μ is

$$\mathbb{P}\left(\mu \notin [L, R] \mid N > N^+\right) \cdot \mathbb{P}\left(N > N^+\right)$$

$$+\mathbb{P}\left(\mu \notin [L, R] \mid N \le N^+\right) \cdot \mathbb{P}\left(N \le N^+\right)$$

$$\le \mathbb{P}\left(N > N^+\right) + \mathbb{P}\left(\mu \notin [L, R] \mid N \le N^+\right)$$

By Lemma 5, the first probability is at most $(1 - \alpha) \cdot \delta$, and Lbound and Rbound are such that the probability of the second term is at most $\alpha \cdot \delta$. The sum of these equals δ, implying that the interval is a valid $(1 - \delta)$ confidence interval, completing the proof. □

Throughout Section 5, we fix $\alpha = 0.99$, giving most of the weight to the confidence interval computation, corresponding to a looser upper bound for N.

Computing CIs for SUM. Now that we have established how to compute CIs for AVG and COUNT, we briefly describe how to combine these two techniques to compute CIs for SUM.Given a $(1 - \frac{\delta}{2})$ confidence interval for COUNT as $[c_\ell, c_r]$ and a $(1 - \frac{\delta}{2})$ confidence interval for SUM as $[g_\ell, g_r]$, union bounding gives $[c_\ell \cdot g_\ell, c_r \cdot g_r]$ as a $(1 - \delta)$ confidence interval for SUM.

Scramble Use and Maintenance under Updates. There exist a few ways to leverage the scramble for analytical queries even for hybrid workloads that additionally involve deletes, updates, and insertions. First, deletes can be handled by simply writing a tombstone to any deleted tuples, and (non-insertion) updates can be handled by simply modifying the updated tuple in-place. In this case, processing the tuples sequentially while ignoring tombstones will remain equivalent to sampling without replacement.

For insertions, there are two options: the first is to leave some holes in the scramble to allow for insertions in random positions, at the cost of increasing space and query time. The second option is to write all updates to a separate smaller "insertion table" that is scanned in full for every query, and then combined with the results of the scramble in a manner which preserves guarantees. This insertion table can then be periodically merged with the scramble and reshuffled. Overall, with some minor extensions inspired from existing big data systems (such as Bigtable [21]) that handle updates by keeping them separate and periodically merging them, scrambles can be readily retrofitted to handle non-read-only workloads.

4.2 Optional Stopping

The techniques discussed in Section 3 describe how to compute high-probability bounds on error given statistics computed from a particular sample of m datapoints. Fixing a sample size ahead of time is oftentimes impractical, since it is usually unknown how

Algorithm 5: The OptStop meta-algorithm

```
Input  :Dataset $\mathcal{D}$ of $N$ values in $[a, b]$, error probability $\delta$
Output :Error bounds that fail to enclose AVG($\mathcal{D}$) with probability $< \delta$

1  $S \leftarrow$ init_state();
2  for $k = 1$ to $\infty$ do
3      for $i = 1$ to $B$ do
4          $v \leftarrow$ sample_without_replacement($\mathcal{D}$);
5          $S \leftarrow$ update_state($S, v$);
6      end
7      $\delta' \leftarrow (6/\pi^2) \cdot (\delta/k^2)$;
8      $L_k \leftarrow$Lbound($S, a, b, \frac{\delta'}{2}$);
9      $R_k \leftarrow$Rbound($S, a, b, \frac{\delta'}{2}$);
10     if should_stop($\left[\max_{j \le k}\{L_j\}, \min_{j \le k}\{R_j\}\right]$) then
11         break;
12     end
13 end
14 return $\left[\max_k\{L_k\}, \min_k\{R_k\}\right]$
```

many samples are needed to ensure CIs that are "just tight enough" to facilitate downstream applications on the part of the user or the system. For example, one approach (e.g. taken by VerdictDB [61]) is to first compute error bounds around an approximate aggregate, and then run an exact query if these bounds are too loose.

Another approach, which we take in this paper, is to continue taking samples until a bound on the error is provably small enough. For this approach, care must be taken to avoid losing guarantees offered by range-based error bounders, since the tighter of two $(1-\delta)$ confidence intervals for a particular aggregate is itself not necessarily a $(1-\delta)$ confidence interval.

Various techniques have been developed for computing *sequentially-valid* confidence intervals as new samples are taken [71, 78, 45, 54, 53, 52]. In addition to techniques for sequential estimation [71] for sequences of i.i.d. random variables from a known family of distributions, various concentration results applicable to AVG which make no distributional assumptions have likewise been derived [78, 45]. Unfortunately, these existing results are derived from variants of Hoeffding's inequality, and therefore suffer from PMA. For the sake of simplicity, we use a much simpler meta-algorithm that can be used in conjunction with any range-based error bounder, including those that leverage our RangeTrim technique, given in Algorithm 5. Although Algorithm 5 requires more samples than the aforementioned techniques when used in conjunction with Hoeffding- or Hoeffding-Serfling-based error bounders, we consider the tradeoff worthwhile due to its generality and simplicity and leave better sequential error bounders to future work.

Analysis of Algorithm 5. Algorithm 5 proceeds in "rounds", with each iteration of the outer loop on line 2 forming a round. During each round, B without-replacement samples are taken and used to incrementally update the state of any range-based error bounder that implements our interface from Section 2.2.2. At the end of each round, confidence intervals are recomputed, with the input error probability δ' decayed "enough" to ensure that the probability of error across *all* rounds is at most δ. If the stopping condition on line 10 is met, then the algorithm terminates; otherwise, it proceeds to the next round, decaying δ' appropriately to control the overall error probability.

We now give a proof of correctness of Algorithm 5.

Theorem 4. *With probability at least* $(1-\delta)$*, the* $\{L_k\}$ *and* $\{R_k\}$ *computed by Algorithm 5 satisfy* AVG$(\mathcal{D}) \in [L_k, R_k]$ *for every* k *in the outer loop. In particular,* AVG$(\mathcal{D}) \in [\max_{k \ge 1} L_k, \min_{k \ge 1} R_k]$ *with probability at least* $(1-\delta)$.

Proof. Denote the δ' used at iteration k with δ_k. Union bounding over rounds, the probability of a mistake is at most

$$\sum_{k \ge 1} \delta_k = \frac{6}{\pi^2} \sum_{k \ge 1} \frac{\delta}{k^2} = \frac{6}{\pi^2} \cdot \frac{\pi^2}{6} \delta = \delta$$

using the identity $\sum_{k \ge 1} \frac{1}{k^2} = \frac{\pi^2}{6}$, completing the proof. □

FastFrame performs I/O at the level of blocks, so instead of computing bounds every B samples as described in the pseudocode of Algorithm 5, we compute bounds after every B block read. In our experiments (Section 5), we set $B = 40000$. We leave development of alternative approaches to future work.

Stopping Conditions for Algorithm 5. Correctness of Algorithm 5 is independent of whether the error bounder uses our RangeTrim technique (please see Appendix A in the appendex for an implementation of our RangeTrim technique in terms of the interface from Section 2.2.2), and it is furthermore independent of stopping condition. We consider several stopping conditions used in our system implementation:

❶ **Desired Samples Taken** ($c \ge m$): If a fixed number of samples are requested, do not use Algorithm 5; instead, terminate query processing once a desired number of tuples contribute to the partial aggregate(s) in the query.

❷ **Sufficient Absolute Accuracy** ($\hat{g}_r - \hat{g}_\ell < \varepsilon$): The interval width is sufficiently small.

❸ **Sufficient Relative Accuracy** ($\max\{\frac{g_r - \hat{g}}{g_r}, \frac{\hat{g} - g_\ell)}{g_\ell}\} < \varepsilon$): The interval width is sufficiently small (relative to the possible correct values implied by the interval).

❹ **Threshold Side Determined** ($v \notin [g_\ell, g_r]$): The interval does not contain some threshold value v, indicating that the true AVG is w.h.p. either less than or greater than the threshold v.

❺ **Top- or Bottom-K Separated**: In a query with multiple groups, the error bounds of the groups with either K smallest or largest aggregates do not intersect those of any of the remaining groups.

❻ **Groups Ordered Correctly**: In a query with multiple groups, the error bounds for each group intersect none of the other groups' error bounds, indicating that the correct ordering of group aggregates has been determined [45].

Different stopping conditions apply to different queries. For example, stopping conditions ❸ and ❹ might be used for the query in Figure 1.

4.3 Active Scanning

For queries with GROUP BYs, different groups may require different numbers of samples to achieve stopping conditions of the types considered in Section 4.2. For simple scans that simply read blocks of the scramble in the order in which they appear, it is impossible to control the relative number of tuples for each group, leading to potential inefficiencies. For example, consider one of the queries in our experiments, F-q2, which selects airlines with average delay above some threshold. This query uses stopping condition ❹ in order to determine when to terminate, since, when this stopping condition has been achieved, it has been determined w.h.p. whether each airline has average delay above or below the threshold. Those groups (airlines) for which the average delay is near $thresh require more samples than those for which the average delay is far from $thresh in order to achieve condition ❹. If these groups are sparse within the scramble, a scan will look at much more data than necessary.

For this reason, we process queries that perform GROUP BYs with an adaptive sampling approach using *active scanning*. Active scanning uses block-based bitmap indexes to efficiently check whether a

Dataset	Size	Tuples	Columns	Replications
Flights	32 GB	606 mil.	5	5×
Taxi	36 GB	679 mil.	4	4×
Police	12 GB	292 mil.	3	72×

Table 3: Dataset descriptions.

block contains tuples for any active group — such groups are marked for processing, and blocks without tuples for any active group are skipped, since they are unlikely to help achieve early termination. The notion of an active group depends on the stopping condition, but in brief, active groups are groups that should be prioritized.

Active Groups for Stopping Conditions. We now describe how we determine active groups, or groups that should be prioritized for sampling, for each of the stopping conditions discussed in §4.2.
❶ (Desired Samples Taken): Under this condition, we consider a group active as long as fewer than the desired m samples have been taken that contribute to that group's aggregate.
❷ (Sufficient Absolute Accuracy): We consider a group active as long as its confidence bounds exceed ε in width; i.e., $\hat{g}_r - \hat{g}_\ell \geq \varepsilon$.
❸ (Sufficient Relative Accuracy): Same as the previous, but a group is active if $\max\{\frac{g_r-\hat{g}}{g_r}, \frac{\hat{g}-g_\ell)}{g_\ell}\} \geq \varepsilon$.
❹ (Threshold Side Determined): A group is active as long as the threshold side has not been determined; i.e., $v \in [g_\ell, g_r]$.
❺ (Top- or Bottom-K Separated):Consider sorting the aggregates for all the groups in increasing order. A group among the top-K is active if its lower confidence bound g_ℓ crosses the midpoint between the aggregate value for the smallest of the top-K and the largest of the bottom $N-K$. Likewise, a group among the bottom $N-K$ is active if its upper confidence bound g_r crosses this midpoint. Analogous statements hold if we consider the separation between the bottom-K and the top $N-K$ as the stopping condition, but with the aggregates sorted in decreasing order.
❻ (Groups Ordered Correctly): A group is active if its interval $[g_\ell, g_r]$ intersects the interval of any other group.

Lookahead. We furthermore accelerate active scanning with a lookahead technique from prior work [56], which we briefly describe here. Instead of checking each block one by one for whether it contains tuples for any active group, active scanning *with lookahead* checks, for each active group, whether each block in a batch of 1024 blocks contains any tuples for that group. By iterating over an entire batch of 1024 blocks for a given active group, bitmaps for the group tend to be in cache more often, making the index lookup operation more efficient. We refer the reader to [56] for more details. In our experiments in Section 5, we set the block size to 64 rows, so a batch of 1024 blocks contains a total of 65536 rows.

5. EMPIRICAL STUDY

In this section, we perform an extensive empirical evaluation of various error bounders and sampling strategies on real data.

5.1 Datasets and Queries

We evaluate various error bounding techniques on publicly available Flights, Taxi, and Police datasets [1, 2, 4]. For Flights, we extract five attributes corresponding to the origin airport, airline, departure delay, departure time, and day of week. To ensure sufficient scale of the data, we perform 5 replications, giving a 32 GiB dataset of 606 million tuples in total. For Taxi, we extract four attributes (for hour of day, passenger count, trip distance, and fare amount) and perform 4 replications. For Police, we extract 3 attributes (for number of violations, officer race, and driver age) and perform 72 replications. These datasets are summarized in Table 3.

Query	Stop When	Parameters Varied
F-q1	(❸) $\max\{\frac{g_r-\hat{g}}{g_r}, \frac{\hat{g}-g_\ell)}{g_\ell}\} < \varepsilon$	\$airport (Figure 6), ε (Figure 7(a))
F-q2	(❹) \$thresh $\notin [g_\ell, g_r]$	\$thresh (Figure 7(b))
F-q3	(❺) bottom-2 separated	\$min_dep_time (Figure 8)
F-q4	(❹) $10 \notin [g_\ell, g_r]$	N/A
F-q5	(❹) $0 \notin [g_\ell, g_r]$	N/A
F-q6	(❺) top-5 separated	N/A
F-q7	(❻) groups ordered	N/A
F-q8	(❺) top-1 separated	N/A
F-q9	(❺) top-1 separated	N/A
T-q1	(❺) top-1 separated	N/A
T-q2	(❺) top-1 separated	N/A
P-q1	(❺) top-1 separated	N/A
P-q2	(❺) top-1 separated	N/A

Table 4: Summary of stopping conditions and template parameters used for queries provided in Figure 5. Template variable arguments shown in blue.

We eliminated rows with "N/A" or erroneous values for any column appearing in one or more of our queries.

Queries and Query Templates. We evaluate our techniques on a diverse set of queries that include various filters and GROUP BY clauses and exercise all the stopping conditions described in Section 4.2 (except conditions ❶ and ❷, which gives similar behavior to condition ❸). The queries themselves are given in Figure 5, and the accompanying stopping conditions are summarized in Table 4. Additionally, several queries are parametrized, in order to reveal interesting data-dependent behavior by varying corresponding parameters. Any query parameters varied are also summarized in Table 4, with parameters shown in blue.

5.2 Experimental Setup

The core of our experiments consists of two ablation studies, intended to evaluate the impact of both our error bounder innovations *and* that of our architectural innovations. In particular, we evaluate various error bounders with and without our RangeTrim technique developed in Section 3, and for the best error bounder (Bernstein+RT), we furthermore evaluate the impact of leaving out features of our active scanning sampling strategy described in Section 4.

We set $\delta = 10^{-15}$ as the default for all queries unless otherwise noted, as we expect users of with-guarantees AQP to desire results that are correct in an *effectively deterministic* manner.

Approaches. We used the following strategies to bound error when running queries in Figure 5:

Error Bounders.

- Bernstein+RT. This uses the empirical Bernstein-Serfling error bounder described in Section 2.3, coupled with our RangeTrim technique described in Section 3, which eliminates PHOS.
- Bernstein. Same as the previous, but without RangeTrim. Bernstein and Bernstein+RT are included to evaluate the impact of an error bounder without PMA.
- Hoeffding+RT. This uses the Hoeffding-Serfling error bounder described in Section 2.3, coupled with our RangeTrim technique described in Section 3, which eliminates PHOS from Hoeffding (but does not fix PMA).
- Hoeffding. Same as the previous, but without RangeTrim.
- Exact. This strawman approach eschews approximation and runs queries exactly, to serve as a simple baseline.

```
# F-q1: avg delay for $airport
SELECT AVG(DepDelay) FROM flights WHERE Origin = $airport
```

```
# F-q2: airlines with avg delay above $thresh
SELECT Airline FROM flights
GROUP BY Airline HAVING AVG(DepDelay) > $thresh
```

```
# F-q3: 2 airlines with min avg delay after $min_dep_time
SELECT Airline FROM flights WHERE DepTime >
      $min_dep_time
GROUP BY Airline ORDER BY AVG(DepDelay) ASC LIMIT 2
```

```
# F-q4: whether ORD has avg delay > 10
SELECT (CASE WHEN AVG(DepDelay) > 10 THEN 1 ELSE 0 END)
FROM flights WHERE Origin = 'ORD'
```

```
# F-q5: airports with negative avg departure delay
SELECT Origin FROM flights
GROUP BY Origin HAVING AVG(DepDelay) < 0
```

```
# F-q6: 5 worst days for afternoon delays across airports
SELECT DayOfWeek, Origin FROM flights
WHERE DepTime > 1:50pm GROUP BY DayOfWeek, Origin
ORDER BY AVG(DepDelay) DESC LIMIT 5
```

```
# F-q7: avg delay by day of week for airline HP
SELECT DayOfWeek, AVG(DepDelay) FROM flights
WHERE Airline = 'HP' GROUP BY DayOfWeek
```

```
# F-q8: origin airport with highest departure delay
SELECT Origin FROM flights GROUP BY Origin
ORDER BY AVG(DepDelay) DESC LIMIT 1
```

```
# F-q9: airline with maximum avg delay
SELECT Airline FROM flights GROUP BY Airline
ORDER BY AVG(DepDelay) DESC LIMIT 1
```

```
# T-q1: Hour with maximum avg distance for trips with 4 passengers
SELECT HourOfDay FROM taxi WHERE passenger_count=4
GROUP BY HourOfDay
ORDER BY AVG(trip_distance) DESC LIMIT 1
```

```
# T-q2: Hour with maximum avg fare for trips with 3 passengers
SELECT HourOfDay FROM taxi WHERE passenger_count=3
GROUP BY HourOfDay
ORDER BY AVG(fare_amount) DESC LIMIT 1
```

```
# P-q1: Officer ethnicity associated with maximum average driver violations per stop
SELECT OfficerRace FROM police
GROUP BY OfficerRace
ORDER BY AVG(violations) DESC LIMIT 1
```

```
# P-q2: Driver age associated with maximum average driver violations per stop
SELECT DriverAge FROM police
GROUP BY DriverAge
ORDER BY AVG(violations) DESC LIMIT 1
```

Figure 5: SQL and semantics for queries. Template parameters are shown in $blue.

We furthermore used the following strategies for sampling when running queries in Figure 5:

Sampling Strategies.

- ActivePeek. This uses the active scanning technique to prioritize groups that are preventing satisfaction of various stopping conditions, along with cache-efficient queries to bitmaps with lookahead (see Section 4.3 for details).
- ActiveSync. This uses active scanning, but processes each block synchronously when deciding whether to read it, incurring high overhead since queries to bitmaps typically result in cache misses.
- Scan. This strategy does not leverage bitmaps in order to decide whether to read a block for active scanning (but may leverage bitmaps for evaluation of whether a block contains tuples that satisfy a fixed predicate, such as the one appearing in F-q1). Without any predicate, this approach simply processes all blocks in the scramble sequentially. Note that the Exact baseline described previously always uses Scan, as only approximate approaches can prune groups.

Environment. Experiments were run on single Intel Xeon E5-2630 node with 125 GiB of RAM and with 8 physical cores (16 logical) each running at 2.40 GHz, although we restrict our experiments to a single thread, noting that our techniques can be easily parallelized. The Level 1, Level 2, and Level 3 CPU cache sizes are, respectively: 512 KiB, 2048 KiB, and 20480 KiB. We ran Linux with kernel version 2.6.32. We report results for data stored in-memory, since the cost of main memory has decreased to the point that many interactive workloads can be performed entirely in-core. Each approximate query was started from a random position in the shuffled data. We found wall clock time to be stable for all approaches, and report times as the average of 3 runs for all methods.

5.3 Metrics

We gather several metrics in order to test two hypothesis: one, that our error bounding strategies in conjunction with our sampling strategies lead to speedups over simpler baselines; and two, that they do so without sacrificing correctness of query results.

Correctness of Query Results. The most important metric is the fraction of queries run that returned correct results, which we discuss briefly here. *Across all methods, all queries, and all parameter settings, results either matched the ground truth determined from an Exact evaluation, or were within error tolerance in the case of F-q1 and F-q7.* This is expected, given that we are consider SSI error bounders with strong probabilistic guarantees, coupled with the fact that our RangeTrim technique and system architecture do not compromise these guarantees. As such, we expect fewer than $\delta = 10^{-15}$ fraction of queries to yield incorrect results, which rounds down to a cool 0.

For the remaining experiments, we focus on the following metrics:

Estimate Error. For a given requested error bound ε supplied to applicable queries, we measure the actual error. The observed error should always fall within the requested error bound.

Wall-Clock Time. Our primary metric evaluates the end-to-end time required for various error bounders and various sampling strategies (where the Exact baseline is included as a "sampling strategy"), across all the queries considered.

Number of Blocks Fetched. We also measure the number of blocks fetched from main memory into CPU cache when using various approaches. This is mainly due to the fact that error bounders incur additional CPU overhead and therefore wall-clock time, with Bernstein and Bernstein+RT incurring the highest overhead, so measuring blocks fetched for these approaches removes this confounding variable by decoupling performance from CPU attributes.

5.4 Results

In this section, we present results of our empirical study.

5.4.1 *Impact of Error Bounder Used*

Summary. Using the Bernstein+RT error bounder resulted in typical speedups of at least 10× over Exact (for 10 out of 13 queries) and Hoeffding (for 9 out of 13 queries), and additionally was almost always on par or better than Bernstein (up to 2× faster).

We evaluate Hoeffding+RT and Bernstein+RT error bounders, along with Hoeffding and Bernstein (to ablate our RangeTrim technique) and an Exact query processor (to ablate any benefits due to approximation) against all the queries in Figure 5, with the resulting time measurements summarized in Table 5.

First we note that all error bounders incur additional overhead — in the case of F-q5 where techniques like Hoeffding and Hoeffding+RT needed to process all the data in order to terminate (due to PMA), they actually ran *more slowly* than Exact. Using Bernstein, which does not suffer from PMA, yielded significant benefits over Exact, Hoeffding, and Hoeffding+RT across all queries. In cases where Hoeffding and Hoeffding+RT showed improvements over Exact, Bernstein amplified these improvements (F-q1, F-q2).

Using RangeTrim in conjunction with both Hoeffding and Bernstein typically led to similar performance, with a few queries exhibiting clearly superior performance (F-q5, F-q6, and F-q3). These queries have the following in common: they all have sparse groups with low selectivity (either because of the large number of groups in the case of F-q5 and F-q3, or because of the restrictive filter in the case of F-q5), and they are all "easy" to approximate, in that none of the groups require too many samples in order to achieve the relevant stopping condition. (F-q8 also has many groups, but some of them require many samples due to a large number of airports with average delay near the max.) This is an ideal condition for Bernstein+RT to show benefit: sparse groups will bottleneck the query, but RangeTrim will achieve termination faster since these sparse groups tend to have fewer outliers than do non-sparse groups. For such bottlenecking sparse groups, the range bounds for the DepDelay column are overly-conservative and dominate the sampling complexity. In this case, Bernstein, which has PHOS, will require twice as many samples for such groups — and since these groups are the bottleneck, it will require roughly twice as much time, an intuition reflected in Table 5.

5.4.2 *Impact of Sampling Strategy Used*

Summary. ActiveSync sampling was typically on par or better than Scan, and ActivePeek sampling was typically on par or better than ActiveSync. ActiveSync significantly outperformed Scan on F-q5, F-q8, and P-q1 (by more than 3×), and ActivePeek significantly outperformed ActiveSync on the same set of queries.

We evaluate the impact of various sampling strategies when used in conjunction with the Bernstein+RT error bounder, the results of which are summarized in Table 6. In some cases (F-q5 and F-q8), the performance of the Scan baseline when used in conjunction with Bernstein+RT was on par with that of the Exact baseline, indicating that some form of block skipping can be crucial for queries with GROUP BYs. When implementing active scanning block by bock as in ActiveSync, the improvement was most significant for F-q5, F-q8, and P-q1, with ActivePeek showing even further improvements for these queries. It is no coincidence that these are the same queries for which Scan performance using approximation is similar to Exact performance. This indicates that there were a few sparse groups preventing termination when Scan is used, which is the very case for which the greatest benefit can be derived from (an efficient implementation of) block skipping.

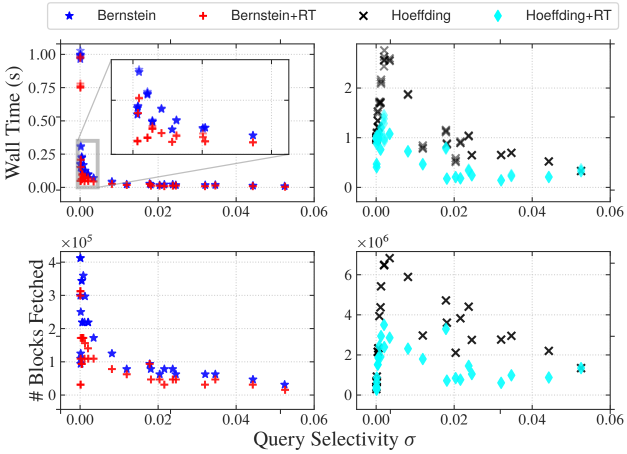


Figure 6: Effect of query selectivity on wall time and blocks fetched, for selectivity determined by varying the origin airport used to filter F-q1[$\varepsilon = .5$].

5.4.3 *Impact of Data and Query Characteristics*

To better understand various data- and query-dependent aspects of our techniques, we now study the effect of varying the parameters supplied to F-q1, F-q2, and F-q3.

Selectivity σ of filter.

Summary. As the fraction of tuples passing F-q1's filter increases, wall clock time and blocks fetched both increase rapidly, then decrease, with RangeTrim giving the most benefit in the case of predicates with intermediate selectivity.

Different Origin attribute values used for filtering F-q1 have different selectivities. By varying the filter attribute value, we reveal interesting behavior impacted by the selectivity of the filter. (We consider selectivity as a number and not a quality, so that larger proportions of tuples satisfy predicates with higher selectivity.) For all four error bounding techniques considered, wall time and blocks fetched are plotted versus query selectivity in Figure 6. Bernstein and Bernstein+RT are plotted separately from Hoeffding and Hoeffding+RT for presentation.

For Hoeffding and Hoeffding+RT bounders, as selectivity increases, both wall-clock time and number of blocks fetched first increase rapidly, then decrease, in a strongly correlated fashion. This is likely because the sparsest filters require examining all the data before terminating, obviating early stopping benefits. After a certain point, however, early termination kicks in, happening more quickly as fewer tuples are filtered. The selectivity threshold for early termination appears to be much lower for Bernstein and Bernstein+RT bounders, which explains why wall-clock time and number of blocks fetched appear to be strictly decreasing with selectivity. The performance gap between techniques with and without RangeTrim generally decreases with increasing selectivity — perhaps because filters with higher selectivity tend to have range bounds that are not as conservative when compared with the a priori range bounds known to hold for the entire column.

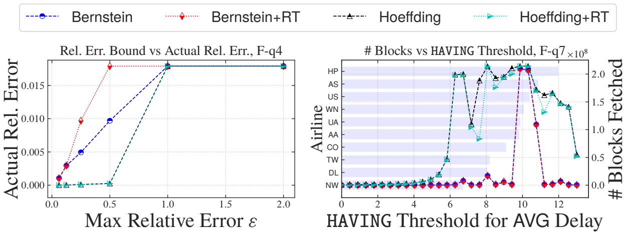
ε for stopping condition ❸.

Query		Avg Speedup over Exact (raw time in (s))			
	Exact (s)	Hoeffding	Hoeffding+RT	Bernstein	Bernstein+RT
F-q1[$airport='ORD',$\varepsilon = .5$]	20.7	58.0× (0.4)	59.6× (0.3)	2129× (0.01)	**2160**× (0.01)
F-q2[$thresh=0]	42.2	233× (0.2)	316× (0.1)	2405× (0.02)	**4683**× (0.01)
F-q3[$min_dep_time=10:50pm]	25.3	1.1× (23.4)	1.7× (15.1)	6.5× (3.9)	**13.8**× (1.8)
F-q4	20.7	12.8× (1.6)	12.7× (1.6)	922× (0.02)	**925**× (0.02)
F-q5	47.7	0.7× (68.8)	1.1× (44.4)	2.2× (21.4)	**4.2**× (11.5)
F-q6	66.4	1.1× (62.4)	1.1× (58.7)	11.2× (6.0)	**16.7**× (4.0)
F-q7	29.8	1.0× (30.0)	1.0× (29.1)	2.6× (11.6)	**2.9**× (10.4)
F-q8	47.7	2.0× (23.3)	1.0× (47.6)	9.1× (5.3)	**9.5**× (5.0)
F-q9	38.2	0.9× (41.0)	1.0× (38.6)	123× (0.3)	**130**× (0.3)
T-q1	32.4	2.2× (14.6)	3.4× (9.5)	17.8× (1.8)	**29.4**× (1.1)
T-q2	39.6	1.6× (24.5)	2.1× (19.0)	19.3× (2.0)	**27.0**× (1.5)
P-q1	20.9	17.8× (1.2)	17.9× (1.2)	311× (0.07)	**314**× (0.07)
P-q2	22.5	11.7× (1.9)	11.3× (2.0)	18.9× (1.2)	**20.8**× (1.1)

Table 5: Summary of average query speedups and latencies for various error bounders.

	Avg Speedup over Scan (time in seconds)		
	Scan	ActiveSync	ActivePeek
F-q5	40.1	2.0× (19.6)	**3.5**× (11.4)
F-q6	4.2	**1.1**× (3.8)	1.1× (3.9)
F-q7	10.3	**1.0**× (10.1)	1.0× (10.4)
F-q8	40.4	3.2× (12.7)	**8.2**× (5.0)
T-q1	1.2	**1.2**× (1.0)	1.1× (1.1)
T-q2	1.6	**1.2**× (1.4)	1.1× (1.5)
P-q2	11.3	7.1× (1.6)	**10.4**× (1.1)

Table 6: Summary of query speedups and latencies for various sampling strategies, restricted to GROUP BY queries that take more than 500ms for Scan with Bernstein+RT.



(a) **(b)**

Figure 7: **(a)**: Effect of requested maximum relative error ε on actual relative error achieved for F-q1. **(b)**: Data required for different HAVING thresholds used in F-q2. The group aggregates for also displayed for comparison.

Summary. For different upper bounds on relative error, the actual relative error in the query result is always within the requested error, for all error bounders applied to F-q1. The achieved relative error drops to 0 more quickly for the more conservative bounders Hoeffding and Hoeffding+RT as the requested error is decreased.

By varying the requested maximum relative error ε for query F-q1, we reveal its impact on the relative error achieved for various error bounders, shown in Figure 7(a). The main takeaways are that, for all error bounders, the achieved relative error generally decreases as the requested error bound decreases, with Hoeffding-based bounders dropping more quickly, as they are more conservative due to PMA.

HAVING threshold for stopping condition ❹.

Summary. HAVING thresholds that are closer to group aggregates require more samples in order to achieve stopping condition ❹, and Hoeffding-based error bounders in particular are more sensitive than Bernstein-based error bounders for the same threshold.

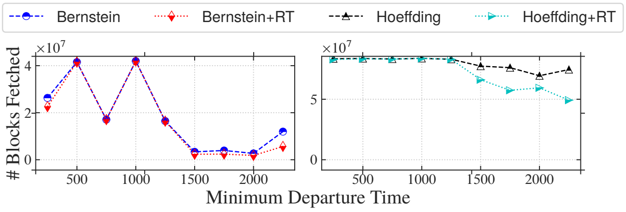


Figure 8: Effect of minimum departure time on blocks fetched for F-q3.

By varying the HAVING threshold used to filter groups / airlines post-aggregate in F-q2 and measuring its effect on the number of blocks fetched for a particular query, we reveal interesting data-dependent behavior impacted by the true aggregates for each airline, depicted in Figure 7(b). This figure also plots the group aggregates using a horizontal bar chart sharing the same x-axis as the HAVING threshold, revealing that it is "harder" to determine which side of the HAVING threshold a given group is if its aggregate is close to the threshold. Indeed, from Figure 7(b), we see that the initial thresholds near 0 are very easy for all groups, allowing for very fast termination. The first spike in number of blocks fetched occurs between 6 and 7, corresponding to the aggregate for airline NW. At and after this point, we see spikes in blocks fetched for both Hoeffding-based and Bernstein-based error bounders whenever the threshold approaches one or more airline aggregate values, although we note that Bernstein-based error bounders appear to be more robust, requiring the threshold to be much closer before they are adversely affected as compared to Hoeffding-based bounders.

Minimum departure time for F-q3.

Summary. As the minimum departure time is increased, the spread of average delay between airlines increases, making it easier to separate the two airlines with the minimum average delays and achieve stopping condition ❺ earlier. At the same time, termination becomes bottlenecked on sparse airlines, increasing the gap between similar bounders with and without RangeTrim.

By varying the minimum departure time `$min_dep_time` in F-q3, we reveal its impact on the number of blocks fetched for various error bounders, shown in Figure 8. This plot exhibits two interesting data-dependent behaviors worth unpacking. First, as the `$min_dep_time` increases, the variance in average delay between different airlines increases, perhaps because some airlines tend to have flights that are delayed more for later flights as compared with other airlines. This makes it easier to achieve stopping condition ❺, since the average delays become more spread out with increasing

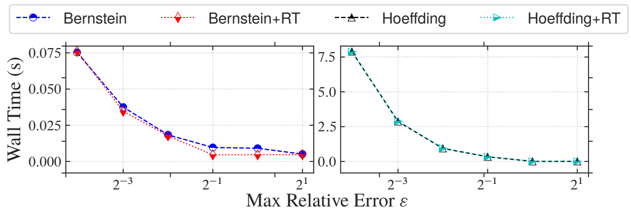


Figure 9: Effect of ε on wall time for F-q1[$origin=ORD]

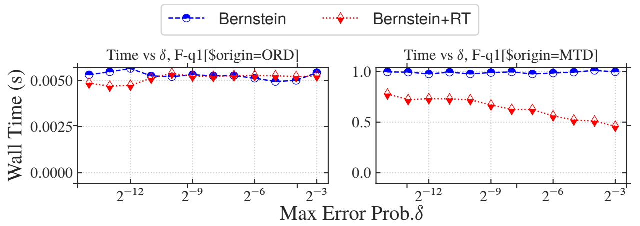


Figure 10: Effect of δ on wall time for F-q1[$origin=ORD] and F-q1[$origin=MTJ], respectively.

minimum departure time, so we observe a decreasing trend in the number of blocks fetched. At the same time, as `$min_dep_time` increases, the selectivity of the various groups decreases. Since all the groups are sparser, the groups for which stopping condition ❺ is bottlenecked are also sparser. Since we have an "easy" query (due to the higher variance between groups) for which sparse groups are bottlenecking termination, we tend to see a bigger performance gap between bounders with and without our RangeTrim technique.

Impact of ε on latency.

> ***Summary.*** Figure 9 demonstrates that latency increases super-exponentially as ε decreases to 0, across all error bounders for F-q1[$origin=ORD].

By varying the maximum relative error bound ε, we reveal its impact on latency for various bounders in F-q1[$origin=ORD]. As depicted in Figure 9, latency increases super-exponentially with decreasing relative error threshold ε (note the log scale on the x-axis). From inspection of formulas for Hoeffding and Bernstein-style bounds, we would expect exponential behavior; the super-exponential behavior can be attributed to the conservative optional stopping procedure, which sacrifices some statistical efficiency. This figure illustrates the importance of being able to leverage other stopping conditions besides that of condition ❸ in the case of queries that do not actually need to make use of the aggregate but merely use it for downstream decision making, since such queries can sometimes tolerate very large relative errors.

Impact of δ on latency.

> ***Summary.*** Figure 10 illustrates that latency is robust to small δ's, with sublinear slowdowns for exponentially decreasing δ.

By varying δ, we reveal its impact on latency for bounders Bernstein and Bernstein+RT on F-q1[$origin=ORD] and F-q1[$origin=MTJ]. In the case of a high-selectivity predicate (i.e., that for F-q1[$origin=ORD]), we see virtually no impact of δ on latency; for this high-selectivity predicate, the query is able to terminate after the first bounds computation for both Bernstein and Bernstein+RT. For the sparser predicate in F-q1[$origin=MTJ], δ does not impact the number of rounds of bounds computation needed for Bernstein, although a larger δ does gradually decrease this quantity for Bernstein+RT, albeit insignificantly considering the exponential increase in δ. Overall, this illustrates that the dependence of each error bounder on $\sqrt{\log{(1/\delta)}}$ translates to query latency that is highly robust in δ, motivating our choice of 10^{-15} as the default δ.

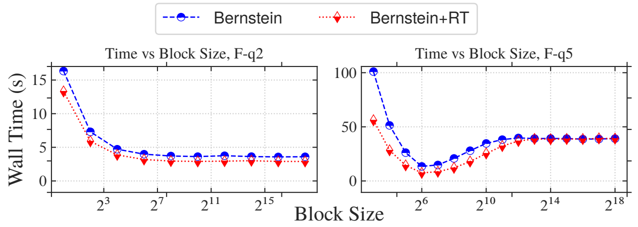


Figure 11: Effect of block size on wall time for F-q2 and F-q5, respectively.

Impact of bock size on latency.

> ***Summary.*** Figure 11 illustrates a quasiconvex relationship between query latency and block size.

By varying the block size, we reveal its impact on latency for F-q2 and F-q5. For both of these queries, smaller block sizes are associated with higher latency due to poorer cache locality and more frequent bounds recomputation. Higher block sizes, however, lose out on benefits from block skipping. This is not an issue for F-q2, which has relatively few groups (and none of which are sparse); for F-q5, however, it impacts latency adversely before leveling off. The level-off in both cases can be attributed to saturating the bitmap indexes, since larger block sizes will be more likely to contain tuples passing each group's filter.

6. RELATED WORK

In this section, we survey related literature and highlight similarities and differences with this work.

Approximate Query Processing (AQP). We survey the AQP literature along two dimensions: first, online versus offline; second, approaches with strong versus asymptotic guarantees.

Online versus Offline AQP. Online sampling-based AQP schemes select samples as queries are issued, contrasted with offline schemes which compute strata ahead of time. Although our approach does perform a shuffle offline, it is nevertheless closer to online schemes, as it uses the scramble to compute samples on the fly as in [56, 64, 75, 76, 31]. Online schemes can use index structures like bitmaps to materialize relevant samples on-the-fly [40, 45, 65, 56], or obey an accuracy constraint for computing predefined aggregates without indices [42, 43]. Offline schemes, on the other hand, materialize samples ahead-of-time [10, 8, 25, 34] based off workload assumptions, sometimes tuning the computed strata as new workload information is available [10, 34].

While we implement our error bounders without PMA or PHOS in the context of a system for online AQP, our core algorithmic techniques are orthogonal to the exact approach, and could be paired with either online or offline schemes.

Sample-size-independent versus Asymptotic Guarantees. Most of the AQP systems from prior work have traditionally leveraged asymptotic error bounders [7, 10, 8, 61], though some have mentioned allowing either approach as an option [40]. Other approaches have leveraged deterministic [37, 63] or concentration-based error bounding techniques [25, 11, 45, 65, 56] under range-based or other very mild assumptions. In some cases, novel asymptotic error bounding techniques have been developed [77, 61, 37] to be used in conjunction with existing systems. Our approach is analogous to these, but instead of basing our techniques on asymptotic methods, we develop error bounding techniques with guarantees independent of sample size, starting from existing concentration-based methods and systematically ameliorating various pathologies.

Of particular note is the work of Agarwal et al. [9], which, as in this paper, recognizes both the pessimism of SSI techniques and the error-proneness of asymptotic techniques. They propose to run a *diagnostic procedure* in conjunction with asymptotic techniques in order to to determine when such techniques are unable to yield accurate answers; however, the diagnostic procedure itself has no guarantees when used for query processing and can give both false positives and false negatives, which we consider unacceptable for the purposes of this paper.

Access Patterns for Informative Samples. A number of techniques have been developed to optimize access to relevant data for analytical queries. Sampling-based approaches [45, 60, 23, 22, 49, 75, 56, 14, 38, 34, 25] attempt to retrieve tuples that will shrink approximation error as quickly as possible. Index structures such as bitmaps [45, 56] or inverted indexes [25] have been employed to facilitate this, or to simply accelerate exact analytical queries by quickly retrieving relevant tuples [74, 19, 46]. We make use of the sampling engine developed in [56], which leverages bitmaps and active scanning to adaptively prioritize different groups in the data while a query is running. While our RangeTrim technique is technically orthogonal to whatever data access method is employed, it demonstrates the most gains over existing error bounding techniques when few samples are needed to terminate. Active scanning is particularly useful for skipping to data needed to terminate when they are sparse.

Another access strategy worth mentioning explicitly comes from from [22] and leverages an outlier index. Outlier indexing [22] works by computing approximate aggregates derived by combining an estimate from the main table and an exact aggregate from the so-called "outlier index", which stores all the rows with outlier values. The benefit of the outlier index is that it shrinks the range of the data from which samples are taken, allowing for faster convergence of approximate answers. One could think of the outlier index as an offline analogy of our own RangeTrim technique. Outlier indexing has some additional limitations that RangeTrim does not have; namely, it cannot be used to facilitate queries with aggregates involving arbitrary expressions, since such expressions can drastically change the set of outlying values. That said, for simple aggregates the two approaches are orthogonal, and could be leveraged together.

Priority sampling [26, 12, 68]is also particularly useful for coping with outliers. If the attribute being aggregated has values $\{w_i\}$, priority sampling computes $\{\alpha_i\}$ (where $\alpha_i \overset{iid}{\sim} \text{Unif}(0,1)$) and estimates $\sum_i w_i$ using the subset of the $\{w_i\}$ with the k largest *priorities*, where the priority for the ith tuple is given by w_i/α_i. While priority sampling applies in the presence of arbitrary filters and can furthermore be modified to allow for computation of AVG aggregates in addition to SUM, it has the drawback that the attribute or expression being aggregated must be known ahead of time (to say nothing of arbitrary expressions), so that the tuples can be sorted in descending order of priority, a limitation our techniques do not have.

Statistical Estimators and Confidence Intervals. The well-known error bounders in statistics and probability leverage asymptotic techniques [29, 67, 39, 28],while those that give strong guarantees independent of sample size beyond Hoeffding's and Serfling's seminal work [41, 66] are relatively more obscure [33, 13]. We surveyed these in Section 2 when we discussed the empirical Bernstein-Serfling error bounder developed by Bardenet et al. [15], which we adapt for use in a database setting with our RangeTrim technique.

7. CONCLUSION AND FUTURE WORK

We categorized existing conservative error bounders in terms of two pathologies, PMA and PHOS, and developed a technique, RangeTrim, for eliminating PHOS from any range-based error bounder. We showed the advantage of using the empirical Bernstein-Serfling bounder in the context of a real system we are developing, FastFrame, that accelerates approximate queries significantly over a Hoeffding-Serfling-based error bounder, which suffers from PMA. We furthermore showed that augmenting this error bounder with our RangeTrim technique leads to an additional 2× in the best case, without ever hurting performance in the worst case. By implementing our distribution-aware techniques in the context of FastFrame, which is aware of practical considerations such as locality, optional stopping, and block skipping in order to prioritize groups that require more samples in order to facilitate early termination, we demonstrate significant speedups (on the order of 10× over both exact processing and traditional techniques based on Hoeffding) without losing guarantees. This suggests a viable path toward practical with-guarantees AQP for workload-agnostic analytics; future work could include, for example, the development of an optimizer that intelligently determines when to leverage traditional data layouts and index structures for exact query processing and when to leverage a scramble for approximate results with exact quality.

8. REFERENCES

[1] Flight Records. http://stat-computing.org/dataexpo/2009/the-data.html, 2009.

[2] NYC Taxi Trip Records. https://github.com/toddwschneider/nyc-taxi-data/, 2015.

[3] Microsoft sql server 2019 documentation: Clr user-defined aggregates – requirements. https://docs.microsoft.com/en-us/sql/relational-databases/clr-integration-database-objects-user-defined-functions/clr-user-defined-aggregates-requirements?view=sql-server-ver15, 2017. Date accessed: 2020-02-27.

[4] WA Police Stop Records. https://stacks.stanford.edu/file/druid:py883nd2578/WA-clean.csv.gz, 2017.

[5] Oracle documentation: Using user-defined aggregate functions. https://docs.oracle.com/cd/B28359_01/appdev.111/b28425/aggr_functions.htm, 2020. Date accessed: 2020-02-24.

[6] Postgresql documentation: User-defined aggregates. https://www.postgresql.org/docs/12/xaggr.html, 2020. Date accessed: 2020-02-24.

[7] S. Acharya, P. B. Gibbons, V. Poosala, and S. Ramaswamy. The aqua approximate query answering system. In *ACM Sigmod Record*, volume 28, pages 574–576. ACM, 1999.

[8] S. Agarwal, A. P. Iyer, A. Panda, S. Madden, B. Mozafari, and I. Stoica. Blink and it's done: interactive queries on very large data. 2012.

[9] S. Agarwal, H. Milner, A. Kleiner, A. Talwalkar, M. Jordan, S. Madden, B. Mozafari, and I. Stoica. Knowing when you're wrong. In *SIGMOD*, pages 481–492, New York, New York, USA, June 2014. ACM Press.

[10] S. Agarwal, B. Mozafari, A. Panda, H. Milner, S. Madden, and I. Stoica. Blinkdb: Queries with bounded errors and bounded response times on very large data. In *EuroSys*, pages 29–42, New York, NY, USA, 2013. ACM.

[11] D. Alabi and E. Wu. Pfunk-h: approximate query processing using perceptual models. In *Proceedings of the 1st Workshop on Human-In-the-Loop Data Analytics*, page 10, 2016.

[12] N. Alon, N. Duffield, C. Lund, and M. Thorup. Estimating arbitrary subset sums with few probes. In *Proceedings of the*

twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, pages 317–325, 2005.

[13] T. W. Anderson. Confidence limits for the expected value of an arbitrary bounded random variable with a continuous distribution function. Technical report, STANFORD UNIV CA DEPT OF STATISTICS, 1969.

[14] B. Babcock, S. Chaudhuri, and G. Das. Dynamic sample selection for approximate query processing. In *SIGMOD*, New York, New York, USA, 2003.

[15] R. Bardenet, O.-A. Maillard, et al. Concentration inequalities for sampling without replacement. *Bernoulli*, 21(3):1361–1385, 2015.

[16] A. C. Berry. The accuracy of the gaussian approximation to the sum of independent variates. *Transactions of the american mathematical society*, 49(1):122–136, 1941.

[17] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[18] G. Casella and R. L. Berger. *Statistical inference*, volume 2. Duxbury Pacific Grove, CA, 2002.

[19] C.-Y. Chan and Y. E. Ioannidis. Bitmap index design and evaluation. In *ACM SIGMOD Record*, volume 27, pages 355–366. ACM, 1998.

[20] T. F. Chan, G. H. Golub, and R. J. LeVeque. Algorithms for computing the sample variance: Analysis and recommendations. *The American Statistician*, 37(3):242–247, 1983.

[21] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems (TOCS)*, 26(2):1–26, 2008.

[22] S. Chaudhuri, G. Das, M. Datar, R. Motwani, and V. Narasayya. Overcoming limitations of sampling for aggregation queries. In *ICDE*, pages 534–542. IEEE, 2001.

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

[24] C. Chen, W. Wang, X. Wang, and S. Yang. Effective order preserving estimation method. In *Australasian Database Conference*, pages 369–380. Springer, 2016.

[25] B. Ding, S. Huang, S. Chaudhuri, K. Chakrabarti, and C. Wang. Sample + seek: Approximating aggregates with distribution precision guarantee. In *SIGMOD*, 2016.

[26] N. Duffield, C. Lund, and M. Thorup. Priority sampling for estimation of arbitrary subset sums. *Journal of the ACM (JACM)*, 54(6):32–es, 2007.

[27] A. Dvoretzky, J. Kiefer, J. Wolfowitz, et al. Asymptotic minimax character of the sample distribution function and of the classical multinomial estimator. *The Annals of Mathematical Statistics*, 27(3):642–669, 1956.

[28] B. Efron. Bootstrap methods: another look at the jackknife. In *Breakthroughs in statistics*, pages 569–593. Springer, 1992.

[29] B. Efron et al. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1):1–26, 1979.

[30] C.-G. Esseen. A moment inequality with an application to the central limit theorem. *Scandinavian Actuarial Journal*, 1956(2):160–170, 1956.

[31] X. Feng, A. Kumar, B. Recht, and C. Ré. Towards a unified architecture for in-rdbms analytics. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 325–336, 2012.

[32] D. Fisher. Incremental, approximate database queries and uncertainty for exploratory visualization. In *2011 IEEE Symposium on Large Data Analysis and Visualization*, pages 73–80. IEEE, 2011.

[33] G. S. Fishman. Confidence intervals for the mean in the bounded case. *Statistics & probability letters*, 12(3):223–227, 1991.

[34] V. Ganti, M.-L. Lee, and R. Ramakrishnan. Icicles: Self-tuning samples for approximate query answering. In *VLDB*, volume 176, 2000.

[35] S. Gupta, S. Purandare, and K. Ramachandra. Aggify: Lifting the curse of cursor loops using custom aggregates. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 559–573, 2020.

[36] P. J. Haas. *Hoeffding inequalities for join-selectivity estimation and online aggregation*. IBM, 1996.

[37] P. J. Haas. Large-sample and deterministic confidence intervals for online aggregation. In *Proceedings. Ninth International Conference on Scientific and Statistical Database Management (Cat. No. 97TB100150)*, pages 51–62. IEEE, 1997.

[38] P. J. Haas and J. M. Hellerstein. Ripple joins for online aggregation. *ACM SIGMOD Record*, 28(2):287–298, 1999.

[39] J. Hájek. Limiting distributions in simple random sampling from a finite population. *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, 5:361–374, 1960.

[40] J. M. Hellerstein, P. J. Haas, and H. J. Wang. Online aggregation. *ACM SIGMOD Record*, 26(2):171–182, jun 1997.

[41] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301):13–30, 1963.

[42] W.-C. Hou, G. Ozsoyoglu, and B. K. Taneja. Statistical estimators for relational algebra expressions. In *Proceedings of the seventh ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 276–287, 1988.

[43] W.-C. Hou, G. Ozsoyoglu, and B. K. Taneja. Processing aggregate relational queries with hard time constraints. In *ACM SIGMOD Record*, volume 18, pages 68–77. ACM, 1989.

[44] C. Jermaine, S. Arumugam, A. Pol, and A. Dobra. Scalable approximate query processing with the dbo engine. *ACM Transactions on Database Systems (TODS)*, 33(4):23, 2008.

[45] A. Kim, E. Blais, A. Parameswaran, P. Indyk, S. Madden, and R. Rubinfeld. Rapid sampling for visualizations with ordering guarantees. *PVLDB*, 8(5):521–532, Jan. 2015.

[46] A. Kim, L. Xu, T. Siddiqui, S. Huang, S. Madden, and A. Parameswaran. Optimally leveraging density and locality for exploratory browsing and sampling. In *Proceedings of the 3rd Workshop on Human-In-the-Loop Data Analytics*, pages 1–7, 2018.

[47] B. C. Kwon, J. Verma, P. J. Haas, and C. Demiralp. Sampling for scalable visual analytics. *IEEE computer graphics and applications*, 37(1):100–108, 2017.

[48] D. Lemire. External-memory shuffling in linear time?, 2010 (accessed August 12, 2020). https://lemire.me/blog/2010/03/15/external-memory-shuffling-in-linear-time/.

[49] F. Li, B. Wu, K. Yi, and Z. Zhao. Wander Join: Online Aggregation via Random Walks. In *Proceedings of the 2016 International Conference on Management of Data*, SIGMOD '16, pages 615–629, New York, NY, USA, 2016. ACM.

[50] K. Li and G. Li. Approximate query processing: What is new and where to go? *Data Science and Engineering*, 3(4):379–397, 2018.

[51] R. F. Ling. Comparison of several algorithms for computing sample means and variances. *Journal of the American Statistical Association*, 69(348):859–866, 1974.

[52] R. J. Lipton and J. F. Naughton. Estimating the size of generalized transitive closures. In *Proceedings of the 15th Int. Conf. on Very Large Data Bases*, 1989.

[53] R. J. Lipton, J. F. Naughton, and D. A. Schneider. *Practical selectivity estimation through adaptive sampling*, volume 19. ACM, 1990.

[54] R. J. Lipton, J. F. Naughton, D. A. Schneider, and S. Seshadri. Efficient sampling strategies for relational database operations. *Theoretical Computer Science*, 116(1):195–226, 1993.

[55] S. Macke, M. Aliakbarpour, I. Diakonikolas, A. Parameswaran, and R. Rubinfeld. Rapid approximate aggregation with distribution-sensitive interval guarantees. Technical report, Available at: https://smacke.net/papers/ddavg.pdf, 2020.

[56] S. Macke, Y. Zhang, S. Huang, and A. Parameswaran. Adaptive sampling for rapidly matching histograms. *Proceedings of the VLDB Endowment*, 11(10):1262–1275, 2018.

[57] P. Massart. The tight constant in the dvoretzky-kiefer-wolfowitz inequality. *The annals of Probability*, pages 1269–1283, 1990.

[58] B. Mozafari. Approximate query engines: Commercial challenges and research opportunities. In *SIGMOD*, pages 521–524. ACM, 2017.

[59] B. Mozafari and N. Niu. A handbook for building an approximate query engine. *IEEE Data Eng. Bull.*, 38(3):3–29, 2015.

[60] F. Olken. *Random sampling from databases*. PhD thesis, University of California, Berkeley, 1993.

[61] Y. Park, B. Mozafari, J. Sorenson, and J. Wang. Verdictdb: Universalizing approximate query processing. In *Proceedings of the 2018 International Conference on Management of Data*, pages 1461–1476, 2018.

[62] A. Pol and C. Jermaine. Relational confidence bounds are easy with the bootstrap. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 587–598, 2005.

[63] N. Potti and J. M. Patel. Daq: a new paradigm for approximate query processing. *Proceedings of the VLDB Endowment*, 8(9):898–909, 2015.

[64] C. Qin and F. Rusu. Pf-ola: a high-performance framework for parallel online aggregation. *Distributed and Parallel Databases*, 32(3):337–375, 2014.

[65] S. Rahman, M. Aliakbarpour, H. K. Kong, E. Blais, K. Karahalios, A. Parameswaran, and R. Rubinfeld. I've seen "enough": Incrementally improving visualizations to support rapid decision making. In *VLDB*, 2017.

[66] R. J. Serfling. Probability inequalities for the sum in sampling without replacement. *The Annals of Statistics*, pages 39–48, 1974.

[67] Student. The probable error of a mean. *Biometrika*, pages 1–25, 1908.

[68] M. Thorup. Confidence intervals for priority sampling. *ACM SIGMETRICS Performance Evaluation Review*, 34(1):252–263, 2006.

[69] L. Valiant. *Probably Approximately Correct: NatureÕs Algorithms for Learning and Prospering in a Complex World*. Basic Books (AZ), 2013.

[70] M. Wainwright. Basic tail and concentration bounds. *URl: https://www. stat. berkeley. edu/.../Chap2_TailBounds_Jan22_2015. pdf (visited 12/31/2017)*, 2015.

[71] A. Wald. *Sequential analysis*. Courier Corporation, 2004.

[72] A. Wald, J. Wolfowitz, et al. Confidence limits for continuous distribution functions. *The Annals of Mathematical Statistics*, 10(2):105–118, 1939.

[73] B. Welford. Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4(3):419–420, 1962.

[74] K. Wu, E. Otoo, and A. Shoshani. Compressed bitmap indices for efficient query processing. *Lawrence Berkeley National Laboratory*, 2001.

[75] S. Wu, B. C. Ooi, and K.-L. Tan. Continuous sampling for online aggregation over multiple queries. In *SIGMOD*, pages 651–662. ACM, 2010.

[76] K. Zeng, S. Agarwal, A. Dave, M. Armbrust, and I. Stoica. G-ola: Generalized on-line aggregation for interactive analysis on big data. In *SIGMOD*, pages 913–918. ACM, 2015.

[77] K. Zeng, S. Gao, B. Mozafari, and C. Zaniolo. The analytical bootstrap: a new method for fast error estimation in approximate query processing. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 277–288, 2014.

[78] S. Zhao, E. Zhou, A. Sabharwal, and S. Ermon. Adaptive concentration inequalities for sequential decision problems. In *Advances in Neural Information Processing Systems*, pages 1343–1351, 2016.

APPENDIX

A. RangeTrim Bounder Pseudocode

Algorithm 6: RangeTrim error bounder

```
Input    : Inner SSI range-based error bounder inner

function init_state()
    return {
        $S_\ell$:  inner.init_state(),
        $S_r$:  inner.init_state(),
        $a'$:  undefined,
        $b'$:  undefined
    };
function update_state($S, v$)
    if $a', b'$ are undefined then
        $S'_\ell \leftarrow S_\ell$;
        $S'_r \leftarrow S_r$;
        $a'' \leftarrow v$;
        $b'' \leftarrow v$;
    else
        $S'_\ell \leftarrow$ inner.update_state($S_\ell, \min(v, b')$);
        $S'_r \leftarrow$ inner.update_state($S_r, \max(v, a')$);
        $a'' \leftarrow \min(a', v)$;
        $b'' \leftarrow \max(b', v)$;
    return {$S_\ell$:  $S'_\ell$,  $S_r$:  $S'_r$,  $a'$:  $a''$,  $b'$:  $b''$};
function Lbound($S, a, b, N, \delta$)
    return inner.Lbound($S.S_\ell, a, S.b', N-1, \delta$);

function Rbound($S, a, b, N, \delta$)
    return inner.Rbound($S.S_r, S.a', b, N-1, \delta$);
```

We give an implementation of our RangeTrim technique in terms of the interface from Section 2.2.2 in Algorithm 6.

B. Handling Arbitrary Expressions

In this paper, we assumed that column c_i was known to lie in some range $[a_i, b_i]$. We then showed how to feed these bounds into our RangeTrim procedure to compute conservative CIs for, e.g., AVG(c_i). In general, however, we may want to compute an aggregate involving an arbitrary expression in terms of several columns. That is, we may want to compute CIs for, e.g., AVG($f(c_1, \ldots, c_n)$). We now show how to do so for a large class of f by optimizing over such f (while using the individual bounds $[a_i, b_i]$ for each column as constraints) in order to compute *derived* range bounds of the form

$$\left[\inf_{c_1,\ldots,c_n} f(c_1,\ldots,c_n),\ \sup_{c_1,\ldots,c_n} f(c_1,\ldots,c_n)\right]$$

Applicable Expressions. To compute a derived lower range bound, we need to be able to either solve or compute a lower bound for the following optimization problem:

$$\begin{aligned} \min_{c_1,\ldots,c_n} \quad & f(c_1,\ldots,c_n) \\ \text{s.t.} \quad & a_i \le c_i \le b_i, \quad \forall 1 \le i \le n \end{aligned}$$

The case for the derived upper range bound is analogous, but with f replaced by $-f$. We show how to compute both lower and upper derived bounds under two kinds of conditions: (i) *the monotonicity condition*; i.e., f is monotone in each c_i, and (ii) *the convexity condition*; i.e., either f or $-f$ is convex. This handles a large number of expressions in practice.

1. Expressions Monotone in each Column. If f is monotone in each column c_i, one simply needs to check whether a_i or b_i gives the smaller (resp. larger) value when computing the lower (resp. upper) bound, and evaluate f on the boundaries for each of these cases.

2. Convex or Concave Expressions. Without loss of generality, we now consider the case of convex f. A large body of existing work focuses on minimizing a convex function subject to convex constraints; please see Boyd et al. [17] for relevant background. In our case, the constraints are all linear (and are sometimes referred to as "box" constraints), and most kinds of convex functions in practice can be optimized efficiently with off-the-shelf software under such constraints, so we do not go into detail here.

Maximizing a f under box constraints is more difficult. Since f is convex, the maximum (and therefore the derived upper range bound we seek) will occur at some set of boundary points; i.e., if $a_i \le c_i \le b_i$, we know that the maximum will occur at one of $c_i = a_i$ or $c_i = b_i$. If we have n columns involved in the expression f, however, we will require evaluating f on all 2^n combinations of boundary points for the constraints. Fortunately, database aggregates over expressions typically do not involve more than 2 or 3 columns, and any $n \le 20$ or so can be handled without trouble.

Example 1. *Suppose the user issues a query to compute AVG($(2c_1 + 3c_2 - 1)^2$) involving columns c_1 and c_2, where we have range bounds $c_1 \in [-3, 1]$ and $c_2 \in [-1, 3]$. The minimum of $(2c_1 + 3c_2 - 1)^2$ subject to these constraints is simply 0, and can be found via quadratic programming. The maximum can be obtained by checking the boundaries $(-3, -1)$, $(-3, 3)$, $(1, -1)$, and $(1, 3)$, and we see that it occurs at $(1, 3)$, for which $(2 \cdot 1 + 3 \cdot 3 - 1)^2 = 100$; thus, the derived range bounds will be $[0, 100]$.*

C. Proof of Theorem 1

In Section 2.2.3, we claimed that the DKW inequality holds for sampling without replacement from a finite population; we now sketch the proof.

Theorem 1. *For any $N > 0$, the DKW inequality applies for sampling without replacement from a finite dataset of size N.*

Proof. Sketch: following the original paper from Wald and Wolfowitz on confidence limits for CDFs [72], it suffices to consider the CDF for mass distributed uniformly at each integer $1, 2, \ldots, N$. For each without-replacement sample size m and each deviation ε, we would like to be able to claim that

$$\mathbb{P}\left(\sup |F_N - \widehat{F}_{N,m}| \ge \varepsilon\right) < \mathbb{P}\left(\sup |F_{N'} - \widehat{F}_{N',m}| \ge \varepsilon\right)$$

for every $N' > N$ — that is, in some sense, the CDF becomes monotonically "harder" to estimate as we increase the dataset size. Unfortunately, this turns out to not be the case, but in fact the claim follows if we merely prove the weaker condition that

$$\mathbb{P}\left(\sup |F_N - \widehat{F}_{N,m}| \ge \varepsilon\right) < \mathbb{P}\left(\sup |F_{N'} - \widehat{F}_{N',m}| \ge \varepsilon\right)$$

for *infinitely many* $N' > N$, implying that, as $N' \to \infty$, the resulting probability to which $\mathbb{P}\left(\sup |F_{N'} - \widehat{F}_{N',m}|\right)$ converges (necessarily bounded by the probability computed in the DKW inequality) is an upper bound for the corresponding probability at every finite N', from which the claim would follow.

We show this via construction: namely, we show that, for every N, m, and ε,

$$\mathbb{P}\left(\sup |F_N - \widehat{F}_{N,m}| \ge \varepsilon\right) < \mathbb{P}\left(\sup |F_{2N} - \widehat{F}_{2N,m}| \ge \varepsilon\right)$$

That is, the CDF becomes monotonically harder to estimate each time we double the dataset size. To show this, we consider two cases. Case 1: if point $2i - 1$ is sampled, then point $2i$ is not sampled, and vice versa for every $i = 1, \ldots, N$. Conditioned on this event, the probability that $\sup|F_{2N} - \widehat{F}_{2N,m}| \geq \varepsilon$ at least the (unconditioned) probability that $\sup|F_N - \widehat{F}_{N,m}| \geq \varepsilon$, since samples at odd indices can only increase the deviation, and samples at even indices cannot decrease it. Case 2: there is at least one i for which points at both indices $2i - 1$ and $2i$ are sampled. Each such point is conceptually similar to reducing m by 1 in the original dataset of size N, but randomly weighting one of the samples by 2 instead of 1. It can be shown that each time this is done, the probability that $\sup|F_N - \widehat{F}_{N,m}| \geq \varepsilon$ increases. □