Extracting Aggregate Answer Statistics for Integration

Zainab Zolaktaf
University of British Columbia
Vancouver, BC, Canada
zolaktaf@cs.ubc.ca

Jian Xu
Microsoft Corporation
Redmond, WA, USA
xujian@microsoft.com

Rachel Pottinger
University of British Columbia
Vancouver, BC, Canada
rap@cs.ubc.ca

ABSTRACT

Aggregate queries in integration contexts often do not have one “true” answer; there can be multiple correct answers for the same aggregate query. This is due to the existence of duplicate or overlapping data points, possibly with different values, across the data sources. Depending on the choice of data source combinations that are used to answer the query, different answers can be generated. Thus, representing the answer to the aggregate query as an *answer distribution* instead of a single scalar value, will allow the users to better understand the range of possible answers.

This work provides a suite of methods for extracting statistics that convey meaningful information about aggregate query answers in heterogeneous integration settings. We focus on the following challenges: 1. determining which statistics best represent an answer’s distribution; and 2. efficiently computing the desired statistics.

Our solution includes the following answer statistics 1. a set of *point estimates* with confidence intervals; 2. a *high coverage interval* that unveils “hot areas” in a distribution; and 3. a *stability score* that measures the impact of source dynamics. We optimize the extraction of the above statistical information by minimizing the sampling load and applying fast approximate algorithms. We verify the effectiveness and efficiency of our methods with empirical studies using real-life and synthetic, scaled data sets.

1. INTRODUCTION

Aggregate queries are fundamental to relational databases. They group sets of data values and calculate informative statistics such as average, median, and sum. They are also important in heterogeneous integration systems [1, 2, 9] where the focus shifts from querying a single database to querying multiple, independently managed, domain heterogeneous databases. The characteristics of heterogeneous information systems make the generation of meaningful aggregate values for heterogeneous information systems significantly more challenging than aggregations in a typical relational database.

D1

Location	Avg Temp	Date
Burnaby	21	10-June-06
Vancouver	19	11-June-06
...	...	...

D2

City	Temp	Date
Burnaby	21	06/10/06
Vancouver	22	06/11/06
Richmond	18	06/12/06
Richmond	18	06/13/06
...	...	...

D3

City	Temp	Date	...	Total Rain
Burnaby	19	10-June-06	...	0.2
Vancouver	17	11-June-06	...	0.0
Surrey	15	11-June-06	...	0.0
Vancouver	20	12-June-06	...	1.4
...	...	...	...	...

D4

Location	Temp	Date	Total Snow	Total Rain
Surrey	15	06/11/06	0.0	0.0
Surrey	19	06/12/06	0.0	1.2
...	...	...	...	...

Figure 1: Climate data from BC weather stations.

Answering aggregate queries in a heterogeneous information system often requires combining sets of data that are segmented across multiple sources. These sources may vary substantially with regard to their schemas and the instances they hold, i.e., semantically related content may be stored in different structures, at different levels of granularity, in different representations, and multiple formats.

There are three levels of heterogeneity in heterogeneous information systems [19]. The first is schema-level. This occurs when there are different schema elements representing the same concept, e.g., one schema may contain a “temp” attribute, while another contains a “temperature” attribute. The second is instance-level heterogeneity. This level requires performing entity resolution to tell if two objects are the same, e.g., that the data for “Vancouver Weather 2006/06/11” in one data source is the same as “Vancouver Weather 06/11/2006” in another. The third is value-level heterogeneity. Heterogeneity at this level deals with the fact that because the sources are independently created and maintained, a given data point can have multiple, inconsistent values across the sources. For example, one source may have the high temperature for Vancouver on 06/11/2006 as 17C, while another may list it as 19C. It is this value-level heterogeneity with which we are concerned throughout this paper. Particularly, we look at the problem of how to handle value-level heterogeneity in aggregations.

To illustrate the problem, consider JIIRP [17], a real-world disaster management project. In JIIRP, data from various sources are combined to simulate the impact of natural disasters. For example, JIIRP assesses weather phenomena and climate data to help plan emergency responses. Figure 1 shows local data sources containing climate data for cities located in British Columbia (BC, Canada). As shown in the figure, the sources differ in terms of their coverage of

10.5441/002/edbt.2015.09

the data instances and attributes. Additionally, data sources D_1, D_2, and D_3 hold different values for the same data point Vancouver on 06/11/2006.

Next, consider the following query, in which the data sources are queried to find the average temperature for months with an average temperature above 20 degrees Celsius:

```
Select Average(Temp), Month(Date), Province(Location)
From SemIS
GROUP BY Province(Location), Month(Date)
HAVING Average(Temp) > 20
```

Applying standard aggregation, however, is incorrect. In particular, standard aggregation would simply return the average of all the points. This is problematic for two reasons: (1) There are some values which are present in more than one source (e.g., Vancouver's temperature on 06/11/2006 is represented in all of sources D_1, D_2, and D_3; taking the simple average will cause Vancouver to be over-represented in the average.) (2) The different sources may have different values for the same conceptual answer (e.g., depending on which source is used, the temperature for Vancouver on 06/11/2006 is either 17, 19, or 22).

The correct aggregation requires using only one value per data point:

$$Average(t) = \frac{\sum_{c=1}^{|\mathcal{C}_{BC}|} \sum_{d=1}^{|\mathcal{D}_m|} t_{c,d}}{|\mathcal{C}_{BC}| * |\mathcal{D}_m|} \tag{1.1}$$

where t is temperature, c represents city, $|\mathcal{C}_{BC}|$ is the number of cities in BC, d represents day, $|\mathcal{D}_m|$ is the number of days in month m. The above aggregation requires 1470 data points (49 cities in BC * 30 days), each of which could have several duplicates across the sources. Depending on the choice of source and value combinations, there can be a whole range of *viable answers*.

For such queries, enumerating all the possible value combinations, and generating the entire set of viable answers as the answer to the aggregate, not only does not scale well (due to combinatorial explosion with regard to the number of data points and possible duplicates), but would still require the users to analyze the results and determine suitable answers.

This problem is not unique to weather data or the JIIRP scenario, for example, [19] examined inconsistency and redundancy at the value-level, in the stock and flight domains on the Deep Web. However, in [19] the authors assumed that there was a single "true" answer, which we do not.

In this paper, we assume meta-information that describes the mappings and bindings between data sources is available [25]. Similar to [19], we focus on value-level heterogeneity. Specifically, we propose a suite of methods for summarizing aggregate query answers in integration settings.

Our previous work [25] created a system where one could ask such aggregate queries. It defined what the possible viable answers were, but then it randomly chose a viable answer. This is inappropriate both since it does not explain the choice, and the aggregate answer would fluctuate upon reprocessing the query. Selecting a best guess answer or a top-K answer set is also inadequate; there is often no global mediator available to choose the value and source combination for the aggregation.

Our solution consists of first estimating the distribution of the viable answers. However, instead of enumerating all the viable answers and computing the full, accurate distribution, we combine a variety of statistical estimations into what we term the *viable answer distribution.* We then efficiently extract statistical summaries of the viable answer distribution to allow the user to better interpret and understand the viable answers. Thus, we contribute the following:

- We define aggregate answers in heterogeneous information systems as a distribution of viable answers computed from different data source and value combinations.

- We provide three summary statistics for the viable answer distribution, consisting of: 1. key point statistics with user defined confidence intervals; 2. high coverage intervals to convey distribution shape information and 3. a stability measure for the aggregation.

- We provide algorithms to efficiently extract the above statistics, including 1. estimating point statistics using sampling and ways to reduce sampling overhead while keeping the confident intervals tight; 2. a fast, greedy algorithm to extract hot areas; and 3. using a probabilistic model to calculate stability scores without simulating source removal. Overall, we can support online extraction of aggregation statistics.

- We describe our empirical study using real-life and synthetic data sets, further verifying our theoretical and algorithmic claims on effectiveness and efficiency.

The paper is organized as follows. We describe preliminary statistical methods applied in our solution in Section 2 and formalize the problem in Section 3. We present the technical details of the distribution estimate process and optimizations in Section 4, the empirical study in Section 5, and review related work in Section 6. We conclude in Section 7.

2. PRELIMINARIES

In this work we use lower case letters for scalars (e.g., a) and typeset the sets (e.g., $\mathcal{A}$); all other variables, including random variables, are denoted by capital letters (e.g., A).

2.1 Bootstrap sampling and bagging

Bootstrap sampling, or bootstrapping, is a resampling technique which combined with Bootstrap aggregating (bagging) [7], can be used to improve the quality of an estimate. The bootstrapping starts with an initial (small) sample set $\mathcal{S}_{alg}$ sampled according to some sampling algorithm alg. This set is then resampled according to alg to obtain a set of bootstrapped sample sets $\mathcal{S}_{boot} = \{\mathcal{B}^i_{boot}\}$ where $i = 1, 2, \ldots, |\mathcal{S}_{boot}|$. Next, an estimator is applied to each set to obtain an ensemble of bootstrapped estimates $\mathcal{E}_{boot} = \{E^i_{boot}\}$. Bagging then approximates a more accurate estimate with tighter confidence intervals by combining (e.g., averaging) this ensemble of estimates. For example, the median value of $\mathcal{E}_{boot}$ can be used as the estimate of the mean of the distribution. We use the above methods to estimate the density of the viable answer distribution.

2.2 Kernel density estimation

Kernel density estimation (KDE) estimates the probability density function (pdf) of a distribution using a sample set drawn from the distribution. We use kernel rather than histogram density estimation due to properties such as smoothness, independence of parameters like bin size, and because KDE often converges to the true density faster. It works as follows: Let the sample set be $\mathcal{S}_{alg} = \{x_i\}$, where $i = 1, 2, \ldots, |\mathcal{S}_{alg}|$. A sample point x_i's contribution to the pdf is measured using a kernel function $K(\frac{x-x_i}{h})$, where h is the bandwidth. After kernels are applied to all x_i's, the pdf is estimated by $f(x) = \frac{1}{|\mathcal{S}_{alg}|h} \sum_1^{|\mathcal{S}_{alg}|} K(\frac{x-x_i}{h})$.

Among the various possible kernel functions, typically Gaussian kernels $K(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$, are used for convenience of theoretical analysis. Note that using a Gaussian kernel makes no assumption that the data adheres to a Gaussian distribution. The bandwidth parameter h controls the localness of a point's impact on the distribution. A large h results in a smooth density function, but is likely to underfit, whereas a small h fits better on the sample points but is likely to over-fit. Selecting an appropriate h value is challenging but there are automatic methods for choosing the value of h [6]. We use KDE to estimate the viable answer density distribution.

2.3 Distance measures for distributions

Several distance measures exist for comparing and quantifying the difference between two distributions p and q. For example, $d_{L_2}(p, q) = \sqrt{\int [p(x) - q(x)]^2 \, dx}$ is the L_2 norm. Also, $d_{Bh}(p, q) = \int \sqrt{p(x) \, q(x)} \, dx$ is the Bhattacharyya distance measure [4] that uses the integral of point-wise product of the two distributions.

In Section 4.4, we quantify the changes of the viable answer distribution under different data source settings. As we will see, the complexity of computing a distance measure largely depends on the mathematical properties of the measure. Our analysis shows that stability scores measured by d_{L_2} and d_{Bh}, as defined above, can be computed efficiently.

3. PROBLEM FORMALIZATION

Data sources in a heterogeneous information system may vary significantly in terms of coverage, quality, and accuracy. Regarding coverage, often a single data source provides information about a subset of the instances and a subset of the object attributes. Therefore, data values relevant to an aggregate query can be segmented across multiple sources in a heterogeneous information system. Furthermore, as in [19], the data sources can contain heterogeneity at three levels: schema-level, instance-level, and value-level. Heterogeneity at the schema-level and instance-levels means semantically related content can be stored using different schema elements and structures, and represented by different instances. At the value-level, depending on the source quality and accuracy, the sources can have inconsistent, or even conflicting data values for the same data points. In this work, similar to [19], we focus on value-level heterogeneity. We assume meta-information that describes the mappings and bindings between data sources is available [25].

Figure 1 demonstrated an example scenario with four data sources $D_1, \ldots, D_4$, and their corresponding data instances. As shown in the figure, the sources held different values for the same data point, e.g., Vancouver on 06/11/2006. As a consequence of the data value overlaps and inconsistencies, the answer to aggregate queries, such as "Sum(Temp)" for specific date ranges and locations, depends on the com-

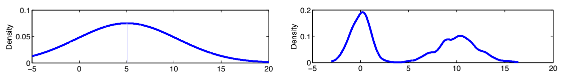


Figure 2: Two distributions with the same mean (5.0) and variance (5.0), but different shapes.

bination of data sources and instances that are selected. Therefore, the answer to the aggregation is a distribution of values, rather than a single scalar value. In order to show which values are under consideration as answers, we use the term "viable answer". While the work in this paper does not depend on the definition, for concreteness, we adopt the definition from [25]:

Definition 1. *[Viable answer]*
Let $\mathcal{D}$ denote a set of data sources answering an aggregation, and let $v = agg(\mathcal{Z})$ be the aggregated value computed from some $\mathcal{Z} \subseteq \mathcal{D}$. Let $\mathcal{V} = \{v_i\}$ be the set of aggregated values from all possible source combinations. A viable answer to an aggregation is a value in the interval $W = [inf(\mathcal{V}), sup(\mathcal{V})]$ that adheres to type restrictions (e.g., integer) where inf and sup are infimum (greatest lower bound) and supremum (smallest upper bound), respectively. □

Note that this definition allows any value in the defined interval, even if it does not correspond to the value produced by any source combination [25]. Furthermore, we assume prior knowledge regarding the coverage, quality and accuracy of the data sources is not available. Therefore, the data sources selected for inclusion have equal importance, but their contribution to the aggregate answer may not be of equal amount. As a reminder, in [25] we randomly chose a viable answer, which is inappropriate, both because it does not explain the choice and the aggregate answer would fluctuate upon reprocessing the query.

Definition 2. *[Viable answer distribution] Let the random variable X be the answer to an aggregate query whose pdf is $f_X^{\mathcal{D}}(x) : W \to \mathbb{R}$. In this notation, the superscript $\mathcal{D}$ denotes the source set used to compute the viable answer set. We refer to $f_X^{\mathcal{D}}$ as the viable answer distribution.*
□

Our objective, in this work, is to efficiently sample the set of viable answers, estimate a viable answer distribution, and report informative statistical summaries that allow the user to better understand the range of viable answers. Typically, the range of possible query answers are conveyed using point estimates such as mean and variance. However, such statistics are not informative when the shape of the density function is unknown; distributions with different shapes can have the same mean and variance but deliver very different information (depicted in Figure 2).

We propose to use the following three statistics as summaries of the viable answer distribution, and the answer to aggregate queries in heterogeneous information systems:

1. **Key point statistics:** Include mean, variance, and skewness of the viable answer distribution. These help users to identify appropriate scalar values for aggregate answers.
2. **High coverage intervals:** Tell where the majority of viable answers can be found. This is particularly useful when the distribution is multi-modal (i.e., the distribution has a pdf with two or more significant peaks.)

3. **A stability measure:** Tells how much the viable answer distribution would change when updates happen or some data sources become unavailable. It helps the heterogeneous information system to decide if a re-processing of an aggregate query is necessary.

The above three statistics communicate semantic information that enable the user to easily interpret and understand the distribution. Specifically, the mean, variance and skewness are standard measures most often desired in describing a distribution. High coverage intervals fill the gap that the point statistics are incapable of providing: "shape" information about a distribution, which is especially useful when the distribution is multi-modal. While the static behavior of the answer distribution is described by the first two statistics, the stability measure captures the update behavior of the distribution.

Furthermore, to focus our work, we assume that the queries that are being asked are aggregate queries. Specifically, we consider sum, average, median, variance, and standard deviation. While our methods may work on other aggregate functions, they were not our focus. We leave removing this restriction as future work.

Algorithm 1: Overall algorithm for extracting statistics

```
input : (User specified) Query $Q$, Data sources $\mathcal{D}$,
        Confidence level $1-\alpha$, Desired coverage $\theta$.
input : (System parameters) Query processor $QP$,
        Initial sample size $|\mathcal{S}_{uniS}|$ (400), #bootstrap
        sample sets $|\mathcal{S}_{boot}|$ (50), Bootstrap sample size
        $|\mathcal{B}^i_{boot}|$(400), Distance measure $d$ ($d_{L_2}$).
output: Mean, variance, skewness point estimates
        $(\mu, \sigma^2, \gamma_1)$, confidence intervals
        $(CI_\mu, CI_{\sigma^2}, CI_{\gamma_1})$;
output: High coverage intervals $(\mathcal{I}, L, C)$;
output: Stability score for $\mathcal{D}$ $Stab_d$.
1  begin
       // Unbiased sampling on viable answers.
2      $\mathcal{S}_{uniS} = UniS(QP(Q), \mathcal{D}, |\mathcal{S}_{uniS}|)$;
       // Bootstrap Sampling.
3      $\mathcal{S}_{boot} = BootstrapSampling(\mathcal{S}_{uniS}, |\mathcal{S}_{boot}|, |\mathcal{B}^i_{boot}|)$;
       // Estimate point statistics.
4      $(\mu, \sigma^2, \gamma_1) = EstPointStatistics(\mathcal{S}_{boot})$;
       // Estimate confidence intervals.
5      $(CI_\mu, CI_{\sigma^2}, CI_{\gamma_1}) = EstCI(\mathcal{S}_{boot}, (\mu, \sigma^2, \gamma_1), \alpha)$;
       // Estimate density function using KDE.
6      $f^{\mathcal{D}}_X = EstDensityFunction(\mathcal{S}_{boot})$;
       // Obtain high coverage intervals.
7      $(\mathcal{I}, L, C)$= $GreedyAlgorithmCIO(f^{\mathcal{D}}_X, \theta)$;
       // Obtain stability score.
8      $Stab_d = Stab(f^{\mathcal{D}}_X, d)$;
9      return Obtained viable answer statistics.
10 end
```

4. EXTRACTING ANSWER STATISTICS

4.1 Overview

Algorithm 1 describes the overall procedure for extracting statistical summaries of the viable answer distribution. The extraction of all three statistics share the uniS sampling and bootstrap sampling steps. In the uniS sampling step (line 2) a set of viable answers are sampled by processing the aggregate query using values from different data sources. This is the most expensive step. The answer set is then bootstrap resampled (line 3). This enables the computation of point statistics such as mean, variance and skewness (line 4) with confidence intervals (line 5). The density estimation process (line 6) is used for finding high coverage intervals (line 7) and for stability analysis (line 8). We perform density estimation on sample sets rather than the entire data set to ensure that the statistics extraction scales well.

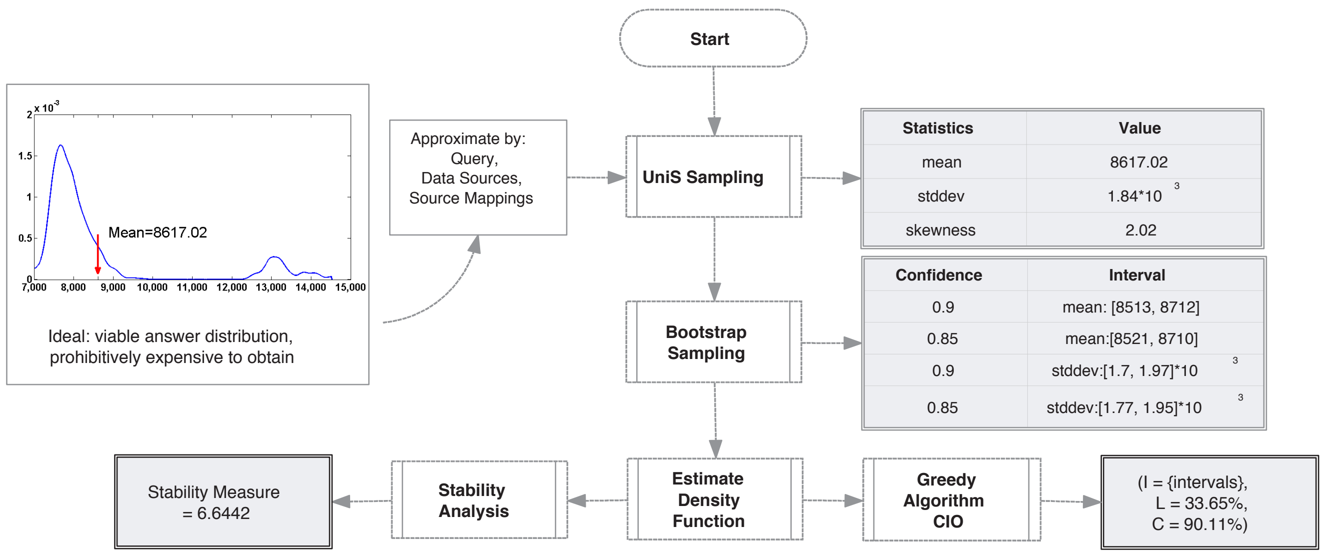
Figure 3 shows the application of Algorithm 1 to obtain answer statistics for the aggregation "Sum(Temp)" over the data sources in Figure 1. Ideally we would prefer to have the exact target viable answer distribution (shown in the top left corner of the figure), however it does not scale well to get this exact distribution. Instead we approximate using the inputs, consisting of the query, data sources, and the mappings between the sources. The algorithm works as follows: UniS sampling samples the data sources to obtain $\mathcal{S}_{uniS}$, a set of viable answer samples, where $|\mathcal{S}_{uniS}| = 400$. Next, this set is resampled to obtain a set of bootstrap sample sets $\mathcal{S}_{boot} = \{\mathcal{B}^i_{boot}\}$, where $|\mathcal{S}_{boot}| = 50$ and $|\mathcal{B}^i_{boot}| = 400$, which are then used to help obtain the 90% and 85% confidence intervals for mean and standard deviation (stddev). After the density function is estimated, Greedy algorithm CIO (Algorithm 2) is used to obtain 3 intervals covering 34% of the range of values, and approximately 90% of the estimated probability distribution. Finally, a stability score of 6.6442 is computed for the query. The outputs of the algorithm are shaded in grey.

In summary, the proposed methods efficiently provide the user with answer statistics that simplify the interpretation of the range of viable answers. This is in contrast to inefficient methods that not only do not scale well, but also require the user to examine and interpret the answers. Sections 4.2 – 4.4 , describe how the three statistics are extracted in detail.

4.2 Sampling and point statistics

We use the term *component* to indicate a data point that an aggregate requires, e.g., in the climate data example in the introduction, a component would be the temperature for Vancouver on 06/11/2006. The process of sampling a viable answer is: (1) find an assignment that determines the use of values from data sources and (2) compute the answer using the chosen assignments. We do not assume prior knowledge regarding the quality, reliability, accuracy, and coverage of the sources. In this context, to ensure correctness, the sampling procedure must select the data sources uniformly and independently to contribute to the aggregate.

We designed a sampling scheme called *uniS* that satisfies the above requirements. Let $\mathcal{D} = \{D_i\}$ where $i = 1, 2, \ldots, |\mathcal{D}|$ denote the data sources, $\mathcal{C}_i$ be the set of components on D_i, and $\mathcal{C}$ be the set of all components needed by the aggregate. UniS starts with an empty component set $\mathcal{T}_0$ and an initial partial aggregate p_0. Step i of uniS uniformly selects one data source D_k, and attempts to add as many components in $\mathcal{C}_k$ to the aggregate, i.e., it updates $\mathcal{T}_i = \mathcal{T}_{i-1} \cup \mathcal{C}_k$ and p_i with partial aggregate computed over component set $\mathcal{A}_i = \{c \mid c \in \mathcal{C}_k, c \notin \mathcal{T}_{i-1}\}$. This process is repeated until $\mathcal{T} = \mathcal{C}$ or all n data sources have been visited. It then computes a final aggregate from the partial


Start
2×10^{-3}
1.5
1
0.5
0
7,000
8,000
9,000
10,000
11,000
12,000
13,000
14,000
15,000
Mean=8617.02
Ideal: viable answer distribution, prohibitively expensive to obtain
Approximate by: Query, Data Sources, Source Mappings
UniS Sampling
Bootstrap Sampling
Estimate Density Function
Greedy Algorithm CIO
Stability Analysis
Stability Measure = 6.6442
(I = {intervals}, L = 33.65%, C = 90.11%)

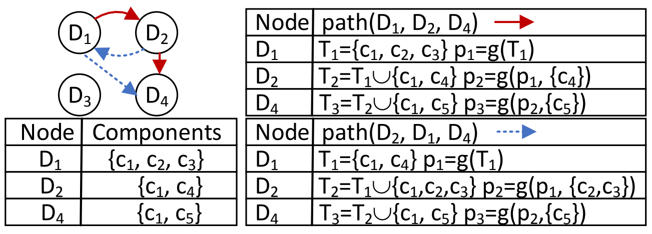
Statistics	Value
mean	8617.02
stddev	$1.84*10^{3}$
skewness	2.02

Confidence	Interval
0.9	mean: [8513, 8712]
0.85	mean:[8521, 8710]
0.9	stddev:$[1.7, 1.97]*10^{3}$
0.85	stddev:$[1.77, 1.95]*10^{3}$

Figure 3: Application of Algorithm 1 to extract statistics for "Sum(Temp)" over data sources in Figure 1.

aggregate and uses it as a viable answer sample. [1]

Figure 4 shows an example scenario where uniS sampling is applied to the sources in Figure 1 for an avg() aggregate over $\mathcal{C} = \{c_1, \ldots, c_5\}$. It shows two different selection paths, $path(D_1, D_2, D_4)$ and $path(D_2, D_1, D_4)$. $\mathcal{T}_i$ is the set of covered components, p_i is the incrementally maintained partial aggregate. The arrows show the different selection paths for $\mathcal{T}_i$ and p_i. For $path(D_1, D_2, D_4)$, the algorithm begins at node D_1 and uses all the components in D_1. The remaining two components c_4 and c_5 are sampled from sources D_2 and D_4, respectively. Note that same component, e.g, c_1, can have different values on different data sources. Hence, using an alternative path, $path(D_2, D_1, D_4)$, yields a different viable answer; since D_2 is visited first, uniS takes both components, c_1 and c_4, from D_2.



Node	Components
D_1	$\{c_1, c_2, c_3\}$
D_2	$\{c_1, c_4\}$
D_4	$\{c_1, c_5\}$

Node	path(D_1, D_2, D_4)
D_1	$T_1=\{c_1, c_2, c_3\}$ $p_1=g(T_1)$
D_2	$T_2=T_1\cup\{c_1, c_4\}$ $p_2=g(p_1, \{c_4\})$
D_4	$T_3=T_2\cup\{c_1, c_5\}$ $p_3=g(p_2,\{c_5\})$

Node	path(D_2, D_1, D_4)
D_1	$T_1=\{c_1, c_4\}$ $p_1=g(T_1)$
D_2	$T_2=T_1\cup\{c_1,c_2,c_3\}$ $p_2=g(p_1, \{c_2,c_3\})$
D_4	$T_3=T_2\cup\{c_1, c_5\}$ $p_3=g(p_2,\{c_5\})$

Figure 4: UniS sampling for selection paths of $\{D_1, D_2, D_4\}$ and $\{D_2, D_1, D_4\}$, where $g = sum()$, and p_i is the incrementally maintained partial aggregate.

Drawing a sample from possibly distributed data sources involves processing an aggregate query with a randomly selected value assignment. Although partial-final aggregates helps to distribute the computational load of each aggregation, applying uniS to draw samples is still a costly operation. Thus, it is desirable to minimize $|\mathcal{S}_{uniS}|$. To this end, we apply bootstrap resampling on the sampled viable answers to improve the confidence for point statistics such as mean and variance. The three parameters: confidence level $1 - \alpha$, confidence interval length denoted by $len(CI)$, and the sampling size $|\mathcal{S}_{uniS}|$ are correlated to affect the computation of confidence intervals. Basically, the larger the samples in the initial set, the higher the accuracy of the point estimates and the confidence intervals. However, to reduce computational costs, we start with a fixed initial sample set and incrementally increase the size of this set (i.e., perform uniS sampling). With each increment, we perform bootstrap resampling and assess the value of $len(CI)$ with the specified α. The procedure ends when the values are satisfactory. Our implementation of bootstrapping uses the standard BC_a [13] method to obtain good quality confidence intervals using small amount of initial samples.

[1]As an example for partial-final aggregate, for final aggregate avg(), the partial aggregate is sum().

4.3 High coverage intervals and optimization

To better understand how viable answers are distributed, we estimate the viable answer distribution (Definition 2) using KDE (Section 2.2). Specifically, we perform density estimation for each bootstrap sample set and use the normalized point-wise mean of all the estimates as the viable answer distribution. Furthermore, we use two methods in addition to standard KDE. The adaptive method described in [6] automatically chooses the value of the bandwidth h depending on the sample set. Bagging (Section 2.1) makes use of the resampled set from bootstrapping. These methods help obtain a density estimation that is both smooth and stable, which is required to extract high coverage intervals and compute stability scores.

We now describe an algorithm for extracting statistics that convey shape information for $f_X^{\mathcal{D}}$, the pdf estimated by KDE.

Definition 3. *[High coverage interval]*
Given an estimated viable answer distribution $f_X^{\mathcal{D}}$, and the viable answer range W, a high coverage interval *is a triple $(\mathcal{I}, L, C)$, where $\mathcal{I} = \{(I_i, C_i) : C_i = \int_{I_i} f_X^{\mathcal{D}}(x)dx$ and $I_i \subseteq W\}$; C_i is the coverage of I_i. $L = \frac{\sum_i |I_i|}{|W|}$ is the fraction*

of the intervals' total length to the viable answer range, and $C = \sum_i C_i$ is the total coverage. □

Definition 4. *[Coverage interval optimization (CIO)] Given a density function $f_X^{\mathfrak{D}}$ for a distribution defined on a finite range, a coverage threshold $0 \le \theta \le 1$, and a constant t representing the number of modes, the CIO problem finds k intervals $I_1, I_2, \ldots, I_k$ where $k \le t$, to minimize $\sum_{i=1}^{k} |I_i|$ subject to $\sum_{i=1}^{k} \int_{I_i} f_X^{\mathfrak{D}}(x)dx \ge \theta$.* □

We further note the following: First, the criterion of minimizing the total interval makes intuitive sense, e.g., the reported high coverage intervals, whether in the weather, the flight, or the stock exchange domain, should be as small as possible. For example, in the flight domain, it is preferable to have shorter rather than larger intervals for the departure time of a certain flight. Second, the coverage threshold θ, conveys the percentage of information covered by the data sources. The desired level can be defined by the user. Third, for single mode distributions (e.g., a Gaussian), the optimal solution is the classical $100\% * \theta$ confidence interval around the mode. The coverage intervals are more useful and deliver important information for multi-modal distributions.

Furthermore, note that k is not a given variable; the CIO problem finds k intervals to minimize the total length of the intervals returned, such that their coverage is above a threshold. Larger coverage is obtained by selecting higher modes. Motivated by this, we propose Algorithm 2 for the extraction of high coverage intervals. The inputs to the algorithm are the pdf $f_X^{\mathfrak{D}}$, the desired coverage θ, and t modes of $f_X^{\mathfrak{D}}$. [2] The algorithm works by greedily picking up new intervals around the modes (lines 5–7) or extending previously picked intervals (lines 9–11) for the highest $t-1$ modes. For the last mode, the interval is set to cover the average amount of the additional coverage needed (lines 17,18). The algorithm returns the obtained high coverage intervals if the desired coverage is met or it has finished searching all the t modes. Our formal analysis of the greedy algorithm relies on the following theorem.

Theorem 4.1 (CIO mode containment property). *If the probability density function $f_X^{\mathfrak{D}}$ of a distribution (1) has t modes and (2) is second-order differentiable everywhere on its range, and the optimal solution for CIO has $k \le t$ intervals, then the largest k modes are contained by the k intervals in the optimal solution.*

Proof If an optimal solution for CIO has k intervals, but the i-th largest mode $(x_i, f_X^{\mathfrak{D}}(x_i)), i \le k$ is not in the optimal solution, then there must exist an interval I_j for which any point $x \in I_j$ satisfies $f_X^{\mathfrak{D}}(x) < f_X^{\mathfrak{D}}(x_i)$; therefore we can construct a new interval around x_i and improve the previous result. □

If the conditions in Theorem 4.1 are satisfied, then the algorithm returns an optimal CIO solution. Otherwise, the greedy algorithm returns an approximation. There are two scenarios for an approximation: (1) the returned intervals may not reach the desired coverage, and (2) optimally choosing the next interval to extend coverage requires picking the I_j that has the minimal $|f_X^{\mathfrak{D}\prime}(x_t^{+/-})|$ (i.e., the largest incremental on coverage).

[2]Because $f_X^{\mathfrak{D}}$ is one-dimensional, the modes are easily computed numerically. We omit details on mode seeking.

Algorithm 2: Greedy algorithm CIO

```
input : Estimated density function $f_X^{\mathfrak{D}}$ over range $W$
input : Desired coverage $\theta$
input : A set $\mathcal{M} = \{(x_i, m_i)\}$ containing $t$ modes of $f_X^{\mathfrak{D}}$
output: High coverage intervals $(\mathcal{I}, L, C)$
1  begin
2      $C \leftarrow 0.0$; $i \leftarrow 1$; array $s$; $\Omega \leftarrow \emptyset$;
3      Sort $\mathcal{M}$ by $m_i$ in descending order;
4      while $C < \theta$, $i \le (t-1)$ do
5          $x_i^- \leftarrow$ largest $x$ s.t. $x < x_i$, $f_X^{\mathfrak{D}}(x) = m_{i+1}$;
6          $x_i^+ \leftarrow$ smallest $x$ s.t. $x > x_i$, $f_X^{\mathfrak{D}}(x) = m_{i+1}$;
7          $s[i] \leftarrow (x_i^-, x_i^+)$; $\Omega \leftarrow \Omega \cup s[i]$; $C \leftarrow \int_\Omega f_X^{\mathfrak{D}}(x)dx$;
           $j \leftarrow 1$;
8          while $C \le \theta$ do
9              $x_j^- \leftarrow$ largest $x$ s.t. $x < x_j$, $f_X^{\mathfrak{D}}(x) = m_{i+1}$;
10             $x_j^+ \leftarrow$ smallest $x$ s.t. $x > x_j$, $f_X^{\mathfrak{D}}(x) = m_{i+1}$;
11             $s[j] \leftarrow (x_j^-, x_j^+)$; $\Omega \leftarrow \Omega \cup s[j]$;
               $C \leftarrow \int_\Omega f_X^{\mathfrak{D}}(x)dx$;
12             $j \leftarrow j + 1$;
13         end
14         $i \leftarrow i + 1$;
15     end
16     if $C \le \theta$ then
17         Find $(x_t^-, x_t^+)$ s.t. $x_t^- < x_t < x_t^+$ ,
           $\int_{x_t^-}^{x_t^+} f_X^{\mathfrak{D}}(x)dx = \frac{1}{t}(\theta - C)$;
18         $s[t] \leftarrow (x_t^-, x_t^+)$; $\Omega \leftarrow \Omega \cup s[t]$; $C \leftarrow \int_\Omega f_X^{\mathfrak{D}}(x)dx$;
19     end
20     Let $\omega_1..\omega_k$ be disjoint intervals s.t. $\bigcup_1^k \omega_i = \Omega$;
21     foreach $\omega_i$ do
22         $C_i = \int_{\omega_i} f_X^{\mathfrak{D}}(x)dx$;
23     end
24     $\mathcal{I} \leftarrow \{(\omega_i, C_i)\}$; $L \leftarrow \sum_1^k |\omega_i|/|W|$;
25     return $(\mathcal{I}, L, C)$;
26 end
```

Obtaining an optimal solution in the above cases requires knowing the first derivative of the density function everywhere, which will bring a substantial cost to the computational overhead. Therefore, we decided to use an approximate answer. Another reason for not pursuing the full optimal solution is that the density function itself is an estimation, and thus has a built-in error. Our empirical study suggests that the greedy algorithm gives a good approximation, and more importantly for query processing, it is fast and scalable.

While the above CIO setting is useful in many situations, the dual of CIO is desired when we are constrained to a pre-determined interval length and asked to return the best possible coverage.

Definition 5. *[Dual of CIO] The dual problem of CIO optimizes the selection of intervals to maximize the coverage. The optimization maximizes $\sum_{i=1}^{k} \int_{I_i} f_X^{\mathfrak{D}}(x)dx$ subject to $\sum_{i=1}^{k} |I_i| = \gamma$, where γ is a user-specified parameter.* □

The greedy algorithm can be easily modified for the dual of CIO by modifying the termination criteria to check if the

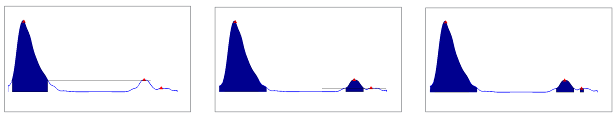

(a) (b) (c)

Figure 5: Finding high coverage intervals (Algorithm 2).

total length of the current set of intervals exceeds length γ and return the size of the covered region.

Figure 5 shows how the high coverage intervals are found for a pdf with 3 modes: the greedy algorithm starts from the highest mode and extends the coverage to lower modes until the coverage of the currently discovered intervals meets the coverage requirement. Eventually 3 intervals are reported.

The returned intervals deliver important information about the aggregation answers. For a fixed coverage θ (e.g., 0.9) and a single mode distribution, if the returned interval length is small, it means that different combinations of sources result in similar aggregation answers. It often indicates that the user can be quite confident of the returned answer. However, we still need to be careful to correctly interpret the result. A small interval length does not necessarily mean that all sources are holding the same value for the same component. It can be the case that the values of some components dominate others. For example, some components are significantly larger than others in a *sum* aggregation.

Additionally, high coverage intervals can be applied in uncertain and probabilistic databases [22]. Such databases represent an attribute as a set of value and probability pairs, $att = \{(A, Pr(A))\}$, where A represents the range of possible values and $Pr(A)$ the probability. High coverage intervals can be used to produce normalized probability measures $att = (I_i, \frac{C_i}{C})$, or simply $att = (I_i, C_i)$, for these databases.

4.4 The stability score for query answers

The point statistics and high coverage intervals provide static information on how viable answers are distributed. However, most heterogeneous information systems, such as PDMSs, are dynamic and data sources may freely leave. One natural question is how to keep aggregate answer statistics up-to-date given that the departure of data sources will affect the aggregate answer distribution. To answer this question, we define the *stability* of the aggregate query.

Stability measures the amount of change caused in the viable answer distribution when some of the sources are removed. It can be quantified as the distance measure between the viable answer distribution without and with some sources removed. Given a number of data sources to be removed r, we randomly remove a set $\mathcal{Q}$ of size r from $\mathcal{D}$ and denote the resulting viable answer distribution by $f_X^{\mathcal{D}\backslash\mathcal{Q}}$. We use the distance measures introduced in Section 2.3 to quantify the difference between the two distributions. Let $\mathcal{S}_{uniS} = \{x_i\}$, be the sampled viable set, with regard to all the sources $\mathcal{D}$, and uniS sampling.

Definition 6. *[Stability score of an aggregation] Let $\mathcal{G}$ be the set of all possible choices for removing r sources from $\mathcal{D}$, and $Pr(\mathcal{Q})$ be the probability of choosing (a particular) $\mathcal{Q}$ with size r. Given $\mathcal{S}_{uniS}$, we define the stability score of the aggregation as*

$$
\begin{aligned}
Stab_d(\mathcal{S}_{uniS}) &\doteq -\log\left(\mathbb{E}[d(f_X^{\mathcal{D}}, f_X^{\mathcal{D}\backslash\mathcal{Q}})]\right) \\
&= -\log\left(\sum_{\mathcal{Q}\in\mathcal{G}} Pr(\mathcal{Q}) d(f_X^{\mathcal{D}}, f_X^{\mathcal{D}\backslash\mathcal{Q}})\right)
\end{aligned}
\tag{4.1}
$$

where $\mathbb{E}$ denotes expectation and d is a distance measure. □

Note that $f_X^{\mathcal{D}}$ is a constant distribution; however, random removal of r sources results in random distributions $f_X^{\mathcal{D}\backslash\mathcal{Q}}$, and subsequently random values for $d(f_X^{\mathcal{D}}, f_X^{\mathcal{D}\backslash\mathcal{Q}})$. We use the expectation of the difference between the two distributions, $\mathbb{E}[d(f_X^{\mathcal{D}}, f_X^{\mathcal{D}\backslash\mathcal{Q}})]$, as the stability measure for the answer distribution. We use the negative logarithmic over the expected distribution difference as the stability score so that a higher value indicates a higher stability score, i.e., that the viable answer distribution is expected to change less against data source changes.

Furthermore, note that we do not need to enumerate or choose $\mathcal{Q}$, and explicitly compute $f_X^{\mathcal{D}\backslash\mathcal{Q}}$; this is just for analysis. The computation of the L_2^2 stability score does not rely on it (Section 2.3).

When additional knowledge regarding the likelihood of the departure of sources is not available, we apply an equal chance assumption on data source removals, where $\Pr(\mathcal{Q}) = 1/\binom{|\mathcal{D}|}{r}$ is constant given r. Another assumption needed for stability analysis is that the number of data sources to be removed is small i.e., $r \ll |\mathcal{D}|$. This assumption is viable, because when the number is large, the system will re-evaluate all the queries regardless of the stability scores. A key benefit to the stability analysis is that it helps prioritize which queries need updating when sources are updated with new values.

Now we describe how to compute the stability score. Given $\mathcal{S}_{uniS} = \{x_i\}$, the sampled viable set, the answer distribution is estimated by $f_X^{\mathcal{D}}(x) = \frac{1}{|\mathcal{S}_{uniS}|h}\sum_{i=1}^{|\mathcal{S}_{uniS}|} K(\frac{x-x_i}{h})$ (Section 4.2). Let $\mathcal{R}_{\mathcal{Q}}$ be the set of sample points that belong to the removed sources $\mathcal{Q}$. To estimate the new distribution, we need to exclude the points in $\mathcal{R}_{\mathcal{Q}}$

$$
\begin{aligned}
f_X^{\mathcal{D}\backslash\mathcal{Q}}(x) &= \frac{1}{(n-|\mathcal{R}_{\mathcal{Q}}|)h}\sum_{x_i\notin\mathcal{R}_{\mathcal{Q}}} K(\frac{x-x_i}{h}) \\
&= \frac{n}{n-|\mathcal{R}_{\mathcal{Q}}|} f_X^{\mathcal{D}}(x) - \frac{1}{(n-|\mathcal{R}_{\mathcal{Q}}|)h}\sum_{x_i\in\mathcal{R}_{\mathcal{Q}}} K(\frac{x-x_i}{h})
\end{aligned}
\tag{4.2}
$$

where $n = |\mathcal{S}_{uniS}|$. The first term with $f_X^{\mathcal{D}}(x)$ is constant given a query, but the second term changes with different choices of $\mathcal{R}_{\mathcal{Q}}$.

We show in Theorem 4.2 that under the equal chance assumption for source removals, stability score can be obtained analytically for the L_2 distance measure.

Theorem 4.2 (L_2 stability score). *Given $\mathcal{S}_{uniS}$, its L_2^2 stability score*

$$
Stab_{L_2}(\mathcal{S}_{uniS}) = -\frac{1}{2}\log\left(\frac{1}{2nh\sqrt{\pi}} * \frac{c_r}{1-c_r}(1-\frac{2}{n(n-1)}\Psi)\right) \tag{4.3}
$$

where $n = |\mathcal{S}_{uniS}|$, h is the bandwidth parameter of the Gaussian kernel, the change ratio is estimated by $c_r = 1 -$

$(1 - \frac{y}{|\mathcal{D}|})^r$, *with y defined as the average number of sources needed for an answer, and the mutual impact factor is* $\Psi = \sum_{i,j} e^{-(x_i-x_j)^2/4h^2}$.

The proof to Theorem 4.2 is in Appendix A. This greatly reduces the computational overhead for the stability score. It eliminates the need of simulating source removal in order to compute a stability score.

We treat the change of viable answers due to data source removals as a random variable. The L_2 stability score is an estimator of the expectation of this random variable. We can also assess its 2nd moment, which gives the variance. To do this, we use the Bhattacharyya distance d_{Bh}, as the distribution difference measure and compute the stability score for the square of the viable answer distribution $\left(f_X^{\mathcal{D}}\right)^2$. Corollary 4.1 shows that this score can also be computed without source removal simulation.

COROLLARY 4.1. *Given $\mathcal{S}_{uniS}$, with n, h, and Ψ as defined in Theorem 4.2, the Bh (Bhattacharyya distance) stability score over the square of the viable answer distribution, is*

$$Stab_{Bh}(\mathcal{S}_{uniS}) = -\log(\frac{1}{2nh\sqrt{\pi}} + \frac{1}{n^2h\sqrt{\pi}}\Psi) \qquad (4.4)$$

Proving the corollary requires the same technique for Theorem 4.2 (in Appendix A). Since the expectation of the density $f_X^{\mathcal{D}\setminus\mathcal{Q}}$ is just $f_X^{\mathcal{D}}$, simple calculation by changing the order of expectation and summation, the result follows.

The stability score measures the likelihood of changes to query answers along with data source availability and updates. It can be used to prioritize the re-evaluation and update of queries, especially in a scenario where multiple continuous queries are managed. Note that the system needs to maintain neither the sampled viable answers nor the density estimation. A priority queue of the stability scores for the continuous queries is sufficient for maintenance.

5. EMPIRICAL STUDY

Dataset: We empirically tested the extraction of aggregate statistics using synthetic and real-life datasets. The synthetic tests allowed us to scale various parameters to verify the observations and predictions made in the analysis. For the real-life data, we experimented with Canadian climate data [8]. This archive contains official weather observations from stations located across Canada. The stations report hourly, daily, and monthly data measurements for attributes including, but not limited to: mean temperature, maximum temperature, total rainfall, total snow, direction of maximum gust, and total precipitation. Not all of the data is quality controlled. Furthermore, the sources may have missing values for some attributes, which typically implies the data had not been observed. In addition to the web interface, the data can be downloaded in XML or CSV format. We used the monthly climate data for the year 2006, from 1672 stations measuring climate data for 104 districts. Tables 1 and 2 summarize our data sets.

Aggregate Query: The query we use in all experiments sums temperature climate data over 500 components from the different data sets in Table 1 ($|\mathcal{C}| = 500$).

Settings: The algorithms were implemented in Matlab version 7.6.0 and the experiments were run on a PC with 2.5GHz Intel Core 2 duo CPU.

Data	Notes	Parameters
D2	A mixture of four Gaussians	$\mu \in$ [10,20], [25,35], [40,50], [55,65], $\sigma = 0.5$, $weight = 12:5:2:1$
D3	A mixture of Gaussians, Cauchy and Gamma	$\mu \in$ [10,20], $\sigma = 1, \infty, 1$
C	(Real-life) Monthly climate data 2006	1672 stations, 104 measuring districts

Table 1: Data set details

Parameter	Symbol	Default
#Data sources	$\lvert\mathcal{D}\rvert$	100
Aggregation size	$\lvert\mathcal{C}\rvert$	500
uniS sample size	$\lvert\mathcal{S}_{uniS}\rvert$	400
#Bootstrap sample sets	$\lvert\mathcal{S}_{boot}\rvert$	50
Bootstrap sample size	$\lvert\mathcal{B}^i_{boot}\rvert$	$\lvert\mathcal{S}_{uniS}\rvert$
Confidence level	$1-\alpha$	90%

Table 2: Parameters in empirical study

5.1 Sampling and bootstrap improvements

Table 3 shows the improvements that are obtained by using the bootstrap method compared to direct inference. In particular, we report improvements in deriving tight confidence intervals for point statistics, and on savings of the required sample size. These experiments were conducted on dataset D2.

In Table 3, the confidence interval length from direct inference denoted by $len(CI_{di})$, is used as the baseline for each individual sampling. We report the maximal and average improvements using an improvement ratio defined as $i_r = \frac{len(CI_{di})}{len(CI_{boot})}$, where $len(CI_{boot})$ is the confidence interval length from bootstrapping. Smaller confidence intervals represent more reliable estimates; thus, greater values of i_r show bootstrapping achieves better performance. We can see that by using bootstrap sampling (with BC_a) with sample sizes of 200 and 400, the average improvement ratio is approximately 2, which shows confidence intervals returned by bootstrapping are half of that guaranteed by direct inference method. They are 3 to 4 times tighter when the sample size is small (200).

Table 3 also shows the savings on the required sample size in order to reach the confidence interval achieved by using the bootstrap method. Similar to the improvement ratio for Table 3, the tighter confidence intervals are translated to the savings on the required size of samples if the same confidence interval length that bootstrapping reports needs to be achieved with direct inference. Hence the saving ratio is defined as $s_r = \frac{|\mathcal{S}_{di}|}{|\mathcal{S}_{uniS}|}$ where $|\mathcal{S}_{di}|$ is the sample size that direct inference needs, and $|\mathcal{S}_{uniS}|$ is the initial sample size that bootstrapping uses. We can see from Table 3 that the average savings on the sample size is about a factor of 4. The results indicate that the confidence interval reported with bootstrap sampling is much tighter than using direct inference especially when the theoretical upper-bound of variance is large.

5.2 High coverage intervals

We present the results of high coverage intervals detected for 4 *sum* aggregations S_1 to S_4 where S_1, and S_2 sum climate data in C and S_3 and S_4 sum values generated in D_3. Figure 7 shows that the viable answer distributions for the 4 aggregates are multi-modal, and when components in

$\lvert \mathcal{S}_{uniS} \rvert$	$1-\alpha$	max i_r	avg i_r	max s_r	avg s_r
200	0.8	4.248	2.556	18.1	7.36
200	0.9	3.309	2.119	10.96	4.84
400	0.8	2.896	2.001	8.39	4.28
400	0.9	2.293	1.655	5.26	2.82

Table 3: Bootstrapping improves confidence intervals and the savings on required sample size.

Fig	Greedy	Optimal	Cover	Greedy/Optimal
a	0.2272	0.2272	85.72%	1.0
b	0.2475	0.2475	85.44%	1.0
c	0.3764	0.2724	73.82%	1.38
d	0.5552	0.5150	92.12%	1.08

Table 4: Approximation ratio of the CIO algorithm

the aggregation are differently distributed, the viable answer distribution has different modes (7 modes in Figure 7.c compared to 8 in Figure 7.d).

By returning intervals in "dense" areas, the intervals cover a small percentage of the range of data (under 25% for S_1, S_2 with 2 modes, 37% for S_3 with 7 modes) to cover the majority of the distribution. The mean of all 4 is in the central flat area; expanding confidence intervals centering at the distribution's mean will result in very large confidence intervals.

Table 4 compares the performance of the greedy algorithm with an optimal method that slices the range of the density function uniformly into 4096 pieces and sums the top t slices that cover the desired probability measure. The greedy approximation is better if the "Greedy/Optimal" value is closer to 1.0 (it is always ≥ 1.0). Note that although the optimal method is more likely to return "tighter' intervals , it does not guarantee the continuity of the returned intervals; thus we used the greedy method in our solution.

5.3 Stability

We designed a simulation process to test the effectiveness and sensitivity of the L_2 stability score. We used the same aggregations as Section 5.2. The simulation worked as follows: we deleted one data source from the 100 sources and drew samples from viable answers that are computable from the remaining 99 sources (e.g., for climate data set, it is 1 out of 104 reporting districts). We recorded the means of the viable answers computed from 99 data sources (103 sources for climate data set). The *deviation maps* in Figure 8 correspond to the four distributions in Figure 7. They show the changes on the sample means when different data sources are disabled. For each circular graph, the center is the mean $\mu^{\mathcal{D}}$ of the viable distribution when no data source is removed; the points represent the means $\mu^{\mathcal{D}\setminus\mathcal{Q}}$ of the viable answer distribution when different data sources are removed. The distance between the data points and the center is defined as $d = \frac{|\mu^{\mathcal{D}\setminus\mathcal{Q}} - \mu^{\mathcal{D}}|}{\mu^{\mathcal{D}}}$.

Comparing the four deviation maps in Figure 8, we can see that answer distributions that have a higher stability score are more "stable", as demonstrated by the fact that the viable distribution means are more densely populated around the center. Note that the L_2 stability does not directly assess the change of distribution means. For example, Figure 2 suggests that two distributions with large differences may have the same mean. We observed this as a consistent trend in our empirical study. While we can confirm that queries with higher L_2 stability scores are more stable, we are not yet able to answer questions like "Will the mean of viable answers shift for more than 10% for a query with score 6.3?" Our suggested use of the stability score is for prioritizing updating of queries by re-evaluating queries with lower stability scores. We plan to focus our future work on deeper evaluation of similarity scores and stability scores when more than one source is removed.

5.4 Processing overhead of operations

Figure 6 reports the execution times of three of the main operations consisting of bootstrap re-sampling, KDE, and greedy CIO algorithm. The time for computing stability scores is negligible and has been discarded (200 iterations required less than a millisecond). Networking overhead times have also been ignored.

As indicated by Figure 6, KDE dominates the processing overhead of the operations. In the experiments, we use 50 bootstrap sample sets with different sizes. The bootstrapping time increases with the sample size but takes less than $3/50 = 60ms$ per run. KDE takes about 5 seconds on 50 sample sets of size 800. The greedy CIO algorithm running time is constant as the sample size increases, since a density function with constant number of 4096 points, is used. We estimate the time needed for computing one viable answer to be $200ms$, which is optimistic since sampling over a distributed hierarchy usually takes up to several seconds when the networking overhead is considered. Therefore, sampling the viable answers dominates the overall time needed for sampling and extracting statistics (e.g., 80 seconds in sampling and 5 seconds to extract statistics). This suggests that our technique is fast and further optimizations should focus on more efficient aggregate computation.

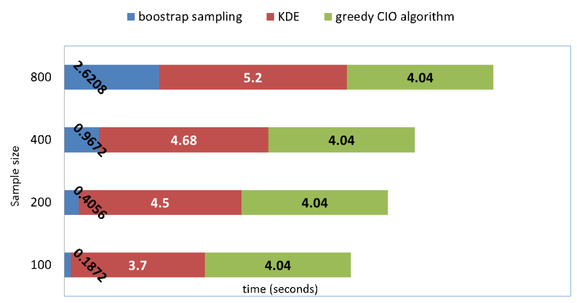
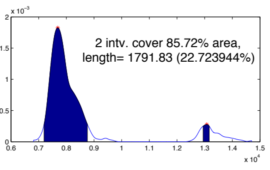


Figure 6: Time breakdown of operations.

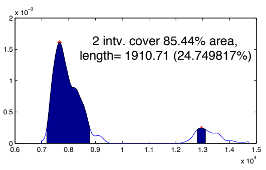
6. RELATED WORK

Data integration is concerned with increasing the coverage of data, and with representing it concisely and accurately [5, 11]. The first objective is typically realized by adding more sources. However, it is argued in [12] that "the more the better" is not always true for integration; obtaining, cleaning, and integrating data can be costly. Furthermore, adding low-quality sources can deteriorate integration quality. Instead, [12] proposes to balance integration cost and gain by selecting a subset of the sources wisely. Our sampling algorithm can be extended with similar ideas.

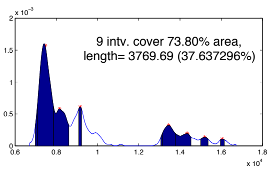
To realize the second objective, concise and accurate data representation, heterogeneity across data sources must be



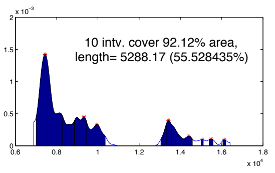
(a) S_1 (Climate Data C)



(b) S_2 (Climate Data C)

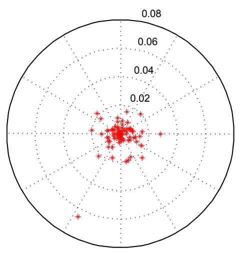


(c) S_3 (Mixture of 3 distributions - dataset D3)

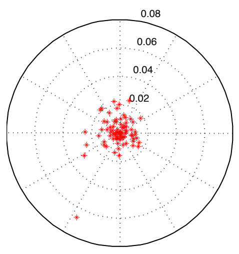


(d) S_4 (Mixture of 3 distributions - dataset D3)

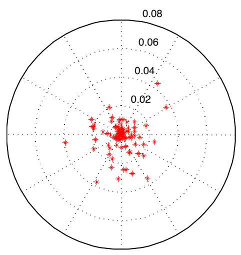
Figure 7: Multi-modal distributions and high coverage intervals



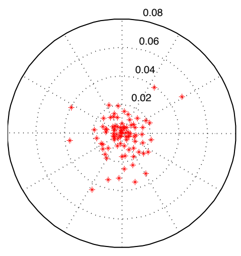
(a) S_1 L_2 score=6.5882



(b) S_2 L_2 score=6.4139



(c) S_3 L_2 score=6.4217



(d) S_4 L_2 score=6.3204

Figure 8: Deviations of empirical means when a single data source is disabled. Numbers indicate the relative distance ($0.02 = 2\%$) from the center. Higher stability scores correspond to figures that have denser distributions around the center.

resolved. As previously mentioned, [19] divides heterogeneity in heterogeneous information systems in three levels: schema (determining which schema elements correspond to each other), instance (determining which data instances correspond to each other), and value (given conflicting values for corresponding instances, choose which one to use). Our work focuses on the third objective. Data fusion [5, 11], also works on the third objective. It assumes a single true value for each component in a data set, and attempts to resolve value conflicts among the sources. In [18] information conflicts are resolved by estimating the reliability of sources and truth values in a joint inference on data with heterogeneous types. In our work, however, do not assume a single true value for components; instead we report a range of possible answers and aim to increase the users understanding and confidence in the reported results. In [19] the truthfulness of data on the Deep web, in particular, the flight and stock domain, is studied. In both domains, large amounts of redundancy and inconsistency at the data value level are observed. Furthermore, state of the art data fusion methods are also compared to resolve conflicts and estimate true values. Semantic ambiguity, out-of-date data, and pure errors are identified as reasons for inconsistent values across the sources. Semantic ambiguity, one the major reasons for value inconsistency, results from different semantics applied by the sources for the attributes they store. For example, one source may compute a statistic of the data over a year-long period, another may compute the same statistic over a half-year period. Both computations are correct with regard to the semantics applied; hence multiple true values are possible [19].

Substantial previous work on combining conflicting data values from multiple sources comes from wireless sensor network research [10, 15, 20, 21]. In a sensor network, sensor motes form an ad-hoc network and collaborate to transmit their sensor probe readings to a centralized repository that is usually beyond the range of a single mote. This is performed by transmitting data in a hierarchical aggregate network rooted at the central repository. Although the hierarchical aggregate network in our case has a lot in common with sensor networks, the operations are essentially different. The aggregate queries we process usually request a (small) part of data maintained in the distributed data sources; in the sensor network case, sensor motes have to upload all their data to the central repository. This results in different optimizations. We face the source combinatorial explosion and use sampling to make estimations, while a sensor network optimizes routing, transmission cost and seeks load balancing among battery powered sensor motes. In [23], the authors described a protocol to adaptively request data from remote sensors over time so as to control the error of approximate answers, while our work focuses on the snapshots of answer distributions when an aggregation is processed. The stability score helps maintain continuous queries but is not a direct approximation measure for a query.

In [14] a method to efficiently compute probability distribution functions in probabilistic databases, is proposed. While using such techniques could eliminate the need for sampling in many cases, in our scenario one of the goals is to reduce the need for data to be gathered from the distributed sources. Since we do not assume that we control the individual sources, we cannot ensure that they have the capabilities proposed in [14].

Research in databases with uncertainty, like our work, covers contexts that do not have one true answer. Various models exist, including uncertain databases [22], possibilistic [24] and probabilistic [16] databases, inconsistent databases [3]. Similarly, in data exchange, aggregate semantics with possible worlds [2] is discussed — there are many possible values that must be considered to find "the answer".

Among the various models, the uncertain database [22] is closest to our setting: it also uses the relational model and the value of an attribute is a discrete distribution with positive probability on a finite set of values. This is similar to our setting, where the values from different data sources are transformed into values in a distribution. In [22] the authors discussed processing aggregate queries on uncertain databases where aggregates uses expectations of values in computation. Their work avoids the combinatorial explosion simply by disallowing exhaustive aggregates and does not give information about the viable answer distribution. In our semantic integration setting, computing the expectation of all component values is impractical as it requires collecting all single values from all data sources. Also, in our stability model, removal of one source will result in invalidating all components' values on that source. Therefore, these techniques cannot be applied to processing aggregates and providing distribution information.

7. CONCLUSION AND FUTURE WORK

This paper proposed our solution for estimating the distribution statistics for viable answers to aggregate queries in integration settings. Our technique extracts essential distribution statistics, returns intervals where aggregate answers have a high chance to be covered, and provides numerical stability scores for aggregate queries. We optimized the computation of the desired statistics using sampling and bootstrapping to minimize the sampling overhead and improved the confidence interval for point estimates of mean and variance. The greedy algorithm computes high coverage intervals quickly and provides good approximations. Our analysis enables the computation of stability scores without simulating source removal. All the optimizations allow the answer distribution estimation techniques described in this paper to be used with a query processing engine.

The stability analysis is the beginning of our investigation on monitoring query answers against changes in the data sources. We will try to establish links between the actual changes on aggregate answers and the stability score values as future work. In addition, the current uniS sampling algorithm assumes equal importance for the sources and samples them uniformly and independently. However, the sources may have different levels of quality and coverage. Future work should consider some notion of provenance. Furthermore, uniS is greedy. While this means that the sampling is not uniform on all viable answers, its speed may be an advantage. In addition, uniS can be fully parallelized as samples are obtained independently. Future work should examine how the algorithm scales when parallelized.

Another future direction, is to make inferences regarding the data and the sources based on the non-normality of the estimated viable answer distribution. In particular, the viable answer distribution can be used to diagnose possible errors in the data. Multi-modal distributions can indicate possible mapping problems in data integration. For example, the second high coverage interval in Figure 7 (a) is caused by combining supposedly cleaned data sets that incorrectly had values in both Fahrenheit and Celsius. Our work can be extended to help automatically detect such errors. Furthermore, using data stratification we can identify homogeneous data sources that apply similar semantics in their computations.

Acknowledgements

We thank Melsa Smith, Mohammad Ghasemi, and Omar AlOmeir for helpful discussions, and the anonymous reviewers for their constructive comments. This work has been funded by NSERC Canada and the NSERC BIN Strategic Network.

8. REFERENCES

[1] F. N. Afrati and R. Chirkova. Selecting and using views to compute aggregate queries (extended abstract). In *ICDT*, pages 383–397, 2005.

[2] F. N. Afrati and P. G. Kolaitis. Answering aggregate queries in data exchange. In *PODS*, 2008.

[3] M. Arenas, L. Bertossi, J. Chomicki, X. He, V. Raghavan, and J. Spinrad. Scalar aggregation in inconsistent databases. *Theoretical Computer Science*, 296:405–434, March 2003.

[4] A. Bhattacharyya. On a measure of divergence between two statistical populations defined by their probability distribution. *Bulletin of the Calcutta Mathematical Society*, 35:99–110, 1943.

[5] J. Bleiholder and F. Naumann. Data fusion. *ACM Comput. Surv.*, 41(1):1:1–1:41, Jan. 2009.

[6] Z. I. Botev, J. F. Grotowski, and D. P. Kroese. Kernel density estimation via diffusion. *Annual of Statistics*, 38(5), 2010.

[7] L. Breiman and L. Breiman. Bagging predictors. In *Machine Learning*, pages 123–140, 1996.

[8] Climate Canada. Canada climate data. `http://climate.weatheroffice.gc.ca/climateData/canada_e.html`, 2010.

[9] S. Cohen, W. Nutt, and A. Serebrenik. Rewriting aggregate queries using views. In *PODS*, pages 155–166, 1999.

[10] A. Deshpande, C. Guestrin, S. R. Madden, J. M. Hellerstein, and W. Hong. Model-driven data acquisition in sensor networks. In *VLDB*, pages 588–599, 2004.

[11] X. L. Dong and F. Naumann. Data fusion: resolving data conflicts for integration. In *VLDB*, pages 1654–1655, 2009.

[12] X. L. Dong, B. Saha, and D. Srivastava. Less is more: selecting sources wisely for integration. In *VLDB*, pages 37–48, 2013.

[13] B. Efron. Better bootstrap confidence intervals. *Journal of the American Statistical Association*, 82(397):pp. 171–185, 1987.

[14] R. Fink, L. Han, and D. Olteanu. Aggregation in probabilistic databases via knowledge compilation. In *VLDB*, pages 490–501, 2012.

[15] N. Jain, D. Kit, P. Mahajan, P. Yalagandula, M. Dahlin, and Y. Zhang. Star: Self-tuning aggregation for scalable monitoring. In *VLDB*, 2007.

[16] T. S. Jayram, S. Kale, and E. Vee. Efficient aggregation algorithms for probabilistic data. In *SODA*, 2007.

[17] Joint infrastructure interdependencies research program—public safety canada. `http://www.civil.engineering.utoronto.ca/staff/professors/eldiraby/News/JIIRP_Project.htm`.

[18] Q. Li, Y. Li, J. Gao, B. Zhao, W. Fan, and J. Han. Resolving conflicts in heterogeneous data by truth discovery and source reliability estimation. In *SIGMOD*, pages 1187–1198, 2014.

[19] X. Li, X. L. Dong, K. Lyons, W. Meng, and D. Srivastava. Truth finding on the deep web: is the problem solved? In *VLDB*, pages 97–108, 2013.

[20] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. In *OSDI*, 2002.

[21] S. Madden, R. Szewczyk, M. Franklin, and D. Culler. Supporting aggregate queries over ad-hoc wireless sensor networks. In *Workshop on Mobile Computing and Systems Applications*, pages 49–58, 2002.

[22] R. Murthy and J. Widom. Making aggregation work in uncertain and probabilistic databases. *TKDE*, 2010.

[23] C. Olston, J. Jiang, and J. Widom. Adaptive filters for continuous queries over distributed data streams. In *SIGMOD*, pages 563–574, 2003.

[24] E. Rundensteiner and L. Bic. Evaluating aggregates in possibilistic relational databases. *DKE*, 7:239–267, 1992.

[25] J. Xu and R. Pottinger. Integrating domain heterogeneous data sources using decomposition aggregation queries. *Information Systems*, 39(0):80 – 107, 2014.

APPENDIX

A. PROOF OF THEOREM 4.2

Proving Theorem 4.2 requires first some background discussion on the product of two Gaussians (Section A.1) and then computing the integral of the square of the density estimation function (Section A.2)

A.1 The product of two Gaussians

Let $f_1(x) \sim \mathcal{N}(\mu_1, \sigma^2)$ and $f_2(x) \sim \mathcal{N}(\mu_2, \sigma^2)$:

$$\begin{aligned} f_1(x)f_2(x) &= \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^2 e^{-\frac{(x-\mu_1)^2+(x-\mu_2)^2}{2\sigma^2}} \\ &= \frac{1}{2\sqrt{\pi\sigma^2}}\mathcal{N}\left(\frac{\mu_1+\mu_2}{2}, \frac{\sigma^2}{2}\right)e^{-\frac{(\mu_1-\mu_2)^2}{4\sigma^2}} \quad \text{(A.1)} \end{aligned}$$

As a special case, let $f(x) = \frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{(x-\mu)^2}{2\sigma^2}} \sim \mathcal{N}(\mu, \sigma^2)$:

$$\begin{aligned} f^2(x) &= \frac{1}{\sqrt{2\pi\sigma^2}}\left[\frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{(x-\mu)^2}{\sigma^2}}\right] \\ &= \frac{1}{2\sqrt{\pi\sigma^2}}\mathcal{N}(\mu, \frac{\sigma^2}{2}) \quad \text{(A.2)} \end{aligned}$$

A.2 The integral of the square of the density function:

With the preparation in Appendix A.1, we can compute the summation of the square of the density estimation function. Recall in KDE using Gaussian kernel, the density function is estimated as $f(x) = \frac{1}{nh}\sum_1^n K(\frac{x-x_i}{h})$, where n is the size of the sample set. We now expand the integral on the square of the density, let $\alpha = \int f^2(x)dx$:

$$\begin{aligned} \alpha &= \int \frac{1}{(nh)^2}\left[\sum_{i=1}^n K(\frac{x-x_i}{h})\right]^2 dx \quad \text{(A.3)} \\ &= \frac{1}{(nh)^2}\int\sum_{i=1}^n K^2(\frac{x-x_i}{h})dx \quad \text{(A.4)} \\ &+ \frac{2}{(nh)^2}\int\sum_{i,j} K(\frac{x-x_i}{h})K(\frac{x-x_j}{h})dx \quad \text{(A.5)} \end{aligned}$$

The kernel function is Gaussian $K(\frac{x-x_i}{h}) = \frac{1}{\sqrt{2\pi}}e^{-\frac{(x-x_i)^2}{2h^2}}$, so $\frac{1}{h}K(\frac{x-x_i}{h}) \sim \mathcal{N}(x_i, h^2)$.

Switching the integral and summation in Equation (A.4), it is is simplified to

$$\frac{1}{2nh\sqrt{\pi}}$$

Similarly, Equation (A.5) is simplified to

$$\frac{1}{n^2h\sqrt{\pi}}\sum_{i,j} e^{-\frac{(x_i-x_j)^2}{4h^2}}$$

We can see that the integral (α) is a function over all the data points.

Now recall the stability analysis using the square d_{L_2} distance. Let $f_X^{\mathcal{D}}(x)$, $f_X^{\mathcal{D}\setminus\mathcal{Q}}(x)$ be the original and augmented density. We want to compute $\mathbb{E}\left[\int\left(f_X^{\mathcal{D}\setminus\mathcal{Q}}(x) - f_X^{\mathcal{D}}(x)\right)^2 dx\right]$. Since $\mathbb{E}[f_X^{\mathcal{D}\setminus\mathcal{Q}}(x)] = f_X^{\mathcal{D}}(x)$, this is the integral of the point-wise variance of the augmented function. It thus equals

$$d_{L_2^2} = \mathbb{E}\left[\int\left(f_X^{\mathcal{D}\setminus\mathcal{Q}}(x)\right)^2 dx\right] - \int\left(f_X^{\mathcal{D}}(x)\right)^2 dx$$

Using the above results on Gaussian square, and letting

$$\Psi = \sum_1^n e^{-(x_i-x_j)^2/4h^2} \quad \text{(A.6)}$$

$$f_X^{\mathcal{D}\setminus\mathcal{Q}} = \frac{1}{h(n-\mathcal{R}_\mathcal{Q})}\sum_1^{n-\mathcal{R}_\mathcal{Q}} e^{-(x_i-x_j)^2/4h^2} \quad \text{(A.7)}$$

$$\mathbb{E}[\mathcal{R}_\mathcal{Q}] = c_r * n \;\; for\, some\, constant\, c_r \quad \text{(A.8)}$$

we have

$$d_{L_2^2} = \frac{1}{2nh\sqrt{\pi}} * \frac{c_r}{1-c_r}(1 - \frac{2}{n(n-1)}\Psi) \quad \text{(A.9)}$$

and thus the stability score formula in Theorem 4.2 follows.

We can see that the distance is related to the viable answer distribution, $f_X^{\mathcal{D}}$, and the average fraction of affected answers when some sources are removed. Also it is easy to verify that when all the data points coincide, the distance is 0, i.e., most stable (the corresponding stability score is ∞).

Now we assess how many answers are likely to be affected when r sources are removed from a total of $|\mathcal{D}|$ data sources. This number relates to the number of data sources required for an answer. Suppose on average we need y (also called the weight) out of $|\mathcal{D}|$ data sources for a viable answer; then an estimate for the fraction that got affected is $c_r = \frac{\binom{|\mathcal{D}|}{y} - \binom{|\mathcal{D}|-r}{y}}{\binom{|\mathcal{D}|}{y}}$. We acknowledge that not all y combinations are answers to the query but this is still a good estimate when information regarding the coverage of the sources is not available. Moreover, the weight itself includes some of this information. The larger the average source coverage for components, the smaller the value of y.

Another way is to estimate the expected number of samples that become invalid when r sources are randomly removed This can be done by simulation with the sample set or using $c_r = 1 - (1 - \frac{y}{|\mathcal{D}|})^r$ which assumes that data sources uniformly contribute to aggregate answers. This completes the proof of Theorem 4.2.

In the proof we use the property of Gaussian distributions which limits the choice of kernel in KDE to Gaussian kernels. We note here that from the perspective of convergence, when the sample sizes increases, using any kernel will converge to the true distribution; thus the expectation of the changes computed here should also converge. Therefore if using any other kernels would make any difference, the divergence is from KDE, but it does not impact the stability analysis.