Sample Debiasing in the Themis Open World Database System

ABSTRACT

Open world database management systems assume tuples *not in* the database still exist and are becoming an increasingly important area of research. We present Themis, the first open world database that automatically rebalances arbitrarily biased samples to approximately answer queries as if they were issued over the entire population. We leverage apriori population aggregate information to develop and combine two different approaches for automatic debiasing: sample reweighting and Bayesian network probabilistic modeling. We build a prototype of Themis and demonstrate that Themis achieves higher query accuracy than a baseline uniform reweighting, an alternative sample reweighting technique, and a variety of Bayesian network probabilistic models while maintaining interactive query response times. We also show that Themis is robust to differences in the support between the sample and population.

ACM Reference Format:

. 2019. Sample Debiasing in the Themis Open World Database System. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Data samples are increasingly easy to access and analyze with the help of websites such as Facebook and Twitter and data repositories such as data.gov, data.world, and kaggle.com. Additionally, data analytic toolkits, like Python, are becoming more mainstream. These two factors have lead to data science becoming tightly coupled with sample analysis.

Modern data scientists, however, face the added challenge that the data samples they seek to analyze are not always an accurate representation of the population they are sampled from. For example, social scientists today study migration patterns from Twitter samples [69], but Twitter users are not a uniform random sample of all people. This phenomenon is known as sample selection bias [21] and is problematic because it can lead to inaccurate analyses.

Correcting this bias, however, is difficult because the sampling mechanism in today's data sources, i.e. the probability of some population tuple being included in the sample, is typically not known. This means common techniques such as the Horvitz-Thompson estimator [10] (see Sec. 4.1) are not applicable.

There is, however, another increasingly available data source scientists leverage for debiasing: population aggregates. Along with the increase in the number of publicly available data samples, there is a recent push for more data transparency and reporting by corporations and governments, e.g. the United State's OPEN Government Data Act passed in 2018 [1] and the InFuse UK aggregate population statistics tool [2]. These reports are often in the form of population aggregate queries. For example, in the FBI's 2017 Internet Crime Report [3], they present a table showing a `GROUP BY, COUNT(*)` aggregate query over crime type, counting the number of victims in each crime type group.

These aggregates can facilitate data debiasing, but the process remains tedious and error prone. There is no general, automatic technique or system for debiasing using aggregates. With the ultimate goal of answering queries approximately over the population, data scientists are forced to manually implement one-off, specialized solutions [69] tailored towards specific datasets, such as census reports [51].

In this paper, we present Themis, which is, to our knowledge, the first open world database management system (DMBS) that automates and encapsulates the debiasing process. The data scientist simply inserts a sample and aggregates and then asks queries, getting approximate results as if the queries were issued on the population. Themis is an open world DBMS as it inherently treats relations as samples and assumes tuples not in the sample could still exist.

This problem of automatically debiasing samples remains unaddressed because DBMSs are traditionally built for closed world business data and do not focus on supporting users working with biased samples. As DBMSs become increasingly used by data scientists [4], however, they need to extend their support to the needs of these new users. Themis takes the first step in that direction.

To achieve our goal, at the heart of our system, we develop and combine two different debiasing techniques: reweighting

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

the sample and learning the probability distribution of the population. The former allows us to more accurately answer heavy hitter queries while the later ensures we can answer queries about tuples that may not exist in the sample.

For sample reweighting, we investigate two different approaches: modifying linear regression and applying an existing aggregate fitting procedure. For learning the probability distribution, we utilize Bayesian networks to build an approximate population probability distribution. The novelty of our system is in not only building two separate debiasing techniques but also combining them into one unified system for query answering. Depending on the query issued, we automatically choose which approach is best using a simple, effective heuristic.

We build a prototype database system called Themis, named after the Greek titan for balance and order who is often seen holding a set of scales and accurately represents our goal of rebalancing data. Themis treats relations as samples and automatically corrects for sample selection bias using population-level aggregates. We evaluate Themis on three datasets to show that Themis is more accurate at answering point queries and top-k queries than standard uniform reweighting, linear regression reweighting (Fig. 11), and a variety of Bayesian network probabilistic approaches (Fig. 10).

In summary, the contributions of this paper are as follows:

- The first open world database system that takes a sample and population aggregates and automatically debiases the data for approximate population query answering (Sec. 3).
- The development and application of debiasing techniques and a novel hybrid approach integrating them (Sec. 4).
- Two optimization techniques for faster preprocessing time: population aggregate pruning and model simplification (Sec. 5).
- Detailed experiments on three datasets showing that Themis achieves a 70 percent improvement in the median error when compared to a naïve scaling approach when asking about heavy hitter tuples (Table 4, Sec. 6). We further show Themis is robust to differences in the support of the sample and the population.

The paper is organized as follows. Sec. 2 gives a motivating example for Themis, Sec. 3 gives the high level model of Themis, Sec. 4 describes our technique in detail, and Sec. 5 discusses our optimization. Finally, Sec. 6 provides experimental results.

2 MOTIVATING EXAMPLE

A data scientist is trying to estimate the number of flights under 30 min in different states of the United States in a year. She has a sample of all flights in the United States skewed towards four major states, but she does not know how badly it is skewed. Further, she has access to how many flights in total leave from each state. She decides to analyze this data and focus only on short flights on either the East or West Coast of the country.

Query	True	Raw	Unif	US State	Themis
CA	7855	2846	28460	7843	7843
FL	2	1	10	3	3
OH	119	1	10	70	70
ME	2	0	0	0	3

Table 1: Query results of the data scientists using the raw sample, a uniformly scaled sample, a state-scaled sample, and Themis.

Being a database user, she ingests the data into a SQL database and prepares to analyze it. As this dataset is a sample, she has three choices for how to prepare her data for analysis: do nothing, ignore skew and uniformly rebalance, use state information to reweight flights based on the number of flights leaving each state. For the second option, she knows there are 7 million flights in the United States per year but only 700,000 in her sample. Therefore, she adds a `weight` attribute to the dataset and gives each tuple a weight of 10, indicating the each tuple in her sample represents 10 tuples in the real world. For the third option, if she knows there are N flights leaving from some state per year but only n leaving that state in her sample, she sets the weight of each flight from that state to be N/n.

Having done preprocessing work, she is ready to ask for the number of quick flights from various states. She starts issuing point queries of the form

```
SELECT SUM(weight) AS num_flights
FROM flights WHERE flight_time <= 30 min
AND origin_state = `<state>';
```

The results of a few state queries are shown in Table 1 where Raw represents option one, Unif represents option two, US State represents option three, and Themis represents our system's answer. Themis and US State use the single aggregate to produces more accurate answers than Raw and Unif because they are correcting for the fact that some flights leaving the four major states are overrepresented in the samples. More importantly, Themis does the re-balancing automatically, which will become time consuming to do manually for more complex aggregates. Themis is also able to answer queries about tuples not in the sample, like ME.

3 THEMIS MODEL

We now describe our data debiasing setup and give an overview of Themis (see Fig. 1). At a high level, Themis uses a sample and population aggregate data to build a model which approximately answers population queries. We use the term *model* because it encapsulates that we use both a reweighted sample and a probabilistic model to answer

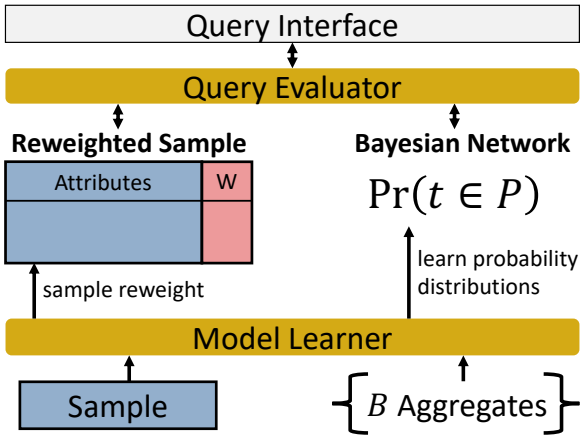

Figure 1: Architecture

queries. Both techniques treat the aggregates as constraints to be satisfied.

Note that the population aggregates do not need to be exact. They may contain errors, be computed at different times, or be purposely perturbed. For example, the 2020 US census will add random noise to their reports to be differentially private [26]. Themis will still treat these aggregates as marginal constraints to be satisfied.

We assume there is a well defined, but unavailable population P of (approximate) size n with m attributes $\mathcal{A} = \{A_1, \ldots, A_m\}$. P is unavailable because either it does not exist (e.g. a dataset of all graduate students in the US) or is not released to the public (e.g. a hospital's private medical data). The active domain of each attribute A_i, of size N_i, is assumed to be discrete and ordered[1].

We assume there is a sample S drawn independently but not uniformly from P of size n_S such that for each tuple $t \in P$, t has probability $\Pr_S(t)$ of being included in S. The subscript S indicates the sampling probability (also called sampling mechanism or propensity score). This probability, however, is not known apriori.

Lastly, we have Γ, a set of results of B aggregate `COUNT(*)` queries of various dimensions computed over the population denoted

$$\Gamma = \{G_{\boldsymbol{\gamma}_i,\text{COUNT}(*)}(P) : i = 1, B\}$$

where $G_{\boldsymbol{\gamma}_i,\text{COUNT}(*)}(P)$ is an aggregate query of dimension d_i; i.e., $\boldsymbol{\gamma}_i \subseteq \mathcal{A}$ (see Example 3.1). Each aggregate query Γ_i returns a set of M_i attribute value-count pairs denoted

$$\Gamma_i = \{(\boldsymbol{a}_{i,k}, c_{i,k}) : k = 1, \ldots, M_i\}$$

where $\boldsymbol{a}_{i,k}$ is the vector of d_i attribute values associated with group k of aggregate i, and $c_{i,k}$ is the group's count.

[1]We support continuous data types by bucketizing their active domains.

We let $\bigcup_{i=1,B} \boldsymbol{\gamma}_i \subseteq \mathcal{A}$, meaning the aggregates may not cover all domain attributes. Note that for each Γ_i, the sum of all counts is approximately the size of P. Further, N_j, the number of distinct values of attribute A_j, can be calculated from any Γ_i if $A_j \in \boldsymbol{\gamma}_i$ or, if there is no such i, S.

When we wish to refer to all the counts and group values of aggregate i separately, we will use $\overline{\Gamma}_i^C$ and $\overline{\Gamma}_i^A$, respectively.

Example 3.1. *Following the example from Sec. 2, assume the population P and sample S are the following sets of domestic flights in the United States.* `date` *is the month and* `o_st` *and* `d_st` *are origin and destination states, respectively.*

$P =$

`date`	`o_st`	`d_st`
01	FL	FL
01	FL	FL
02	FL	NY
01	NC	FL
02	NC	NY
02	NC	NY
02	NC	NY
01	NY	FL
01	NY	NC
02	NY	NY

$S =$

`date`	`o_st`	`d_st`
01	FL	FL
01	FL	FL
02	NC	NY
01	NY	NC

Let $\Gamma = \{\Gamma_1, \Gamma_2\}$ with $d_1 = 1$ and $d_2 = 2$ be the following two aggregate queries.

$\Gamma_1 = G_{\texttt{date},\texttt{COUNT(*)}}(P) = \{([01], 5), ([02], 5)\}$

$\Gamma_2 = G_{\texttt{o\_st},\texttt{d\_st},\texttt{COUNT(*)}}(P) =$

$\{([FL, FL], 2), ([FL, NY], 1), ([NC, FL], 1),$

$([NC, NY], 3), ([NY, FL], 1), ([NY, NC], 1), ([NY, NY], 1)\}.$

In this case, $n = 10$, $B = 2$, $\overline{\Gamma}_1^C = [5, 5]$, $\overline{\Gamma}_1^A = \begin{bmatrix} 01 \\ 02 \end{bmatrix}$, $\overline{\Gamma}_2^C = [2, 1, 1, 3, 1, 1, 1]$, and $\overline{\Gamma}_2^A = \begin{bmatrix} FL & FL \\ FL & NY \\ NC & FL \\ NC & NC \\ NY & FL \\ NY & NC \\ NY & NY \end{bmatrix}$. The ordering of the rows of $\overline{\Gamma}_i^C$ and $\overline{\Gamma}_i^A$ does not matter, as long as it is consistent.

We are given a user query Q over the population. While Q can be any SQL query, we focus on simple point and top-k queries to study the improvement in accuracy by using Themis. As we do not have P, we need to estimate $Q(P)$ given S and Γ. To do so, we build a model $\mathcal{M}(\Gamma, S)$ such that $Q(\mathcal{M}(\Gamma, S))$ is an approximate answer to $Q(P)$.

4 DATA DEBIASING

We are now ready to present how Themis builds $\mathcal{M}(\Gamma, S)$. Themis has two components: a reweighted sample and a

probabilistic model. We present each technique and then describe how Themis merges them into a unique hybrid approach to answer $Q(P)$ approximately.

4.1 Sample Reweighting

In sample reweighting, each tuple $t \in S$ gets assigned a weight $w(t)$ representing the number of times t appears in P. Queries on P get transformed to run on weighted tuples by, for example, translating `COUNT(*)` to be `SUM(weight)`. If the sampling mechanism, $\Pr_S(t)$, is known, we can use the Horvitz-Thompson estimator which reweights each tuple by $1/\Pr_S(t)$ [21, 50].

The challenge is that we do not have the sampling mechanism. A simple approach is to perform uniform reweighting by setting $w(t)$ to be $|P|/|S|$. As we show in Sec. 6, when the sample is biased, this achieves low accuracy. To correct for the bias, we present two solutions for learning $w(t)$ using the sample S and aggregate information Γ. The first technique is to adapt linear regression, and the second is to apply Iterative Proportional Fitting (IPF) [37, 45].

4.1.1 Linear Regression Reweighting. Following propensity score research [9, 49, 58], we make the same assumption that a tuple's weight depends on the attributes of t. In particular, we assume the weight of a tuple is a linear combination of its attributes. In other words, if $\boldsymbol{t}^{0/1}$ represents the one-hot encoded tuple t, then $w(t) = \boldsymbol{\beta} \cdot \boldsymbol{t}^{0/1}$ where $\boldsymbol{\beta}$ is a vector of weights. Note that for the rest of this section, we use m to refer to the number of attributes covered by the aggregates and only use those attributes for learning the weight.

Example 4.1. *Continue with Example 3.1. The one-hot encoded version of the first tuple in S,* $(01, FL, FL)$*, is*

$$\begin{matrix} d_{01} & d_{02} & o_{FL} & o_{NC} & o_{NY} & d_{FL} & d_{NC} & d_{NY} \end{matrix}$$
$$\begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}.$$

To solve for $\boldsymbol{\beta}$, S gets represented by a $n_S \times m^{0/1}$ matrix, $\boldsymbol{X}_S$, where $m^{0/1} = \sum_{i=1}^{m} N_i + 1$ (the plus one is because we use the standard formulation of adding a column of ones to represent the intercept). $\boldsymbol{y}$ is $[\overline{\Gamma}_1^C, \ldots, \overline{\Gamma}_B^C]$, a vector of all the aggregate queries' count values; i.e.

$$\boldsymbol{y} = \begin{bmatrix} c_{1,1} & \cdots & c_{1,M_1} & \cdots & c_{B,1} & \cdots & c_{B,M_B} \end{bmatrix}^T.$$

Let $\boldsymbol{X}$ be the matrix product of $\boldsymbol{G}^{0/1}\boldsymbol{X}_S$ where $\boldsymbol{G}^{0/1}$ is a 0/1 matrix with $\sum_{i=1}^{B} M_i$ rows and n_S columns (see Example 4.2). $\boldsymbol{G}^{0/1}$ is an incidence matrix where row r and column c is 1 if row c of $\boldsymbol{X}_S$ participates in the rth group by result; i.e., if the rth attribute value from $\bigcup_i \overline{\Gamma}_i^A$ is in row c of S. We then solve

$$[\boldsymbol{G}^{0/1}\boldsymbol{X}_S]\boldsymbol{\beta} = \boldsymbol{y}. \tag{1}$$

In the case an entire row of $\boldsymbol{G}^{0/1}\boldsymbol{X}_S$ is all zeros, which happens with missing values in S, we drop that row and its associated value in $\boldsymbol{y}$.

Departing from standard linear regression solving techniques, we solve Eq. 1 using a constrained least squares formulation to constrain $\boldsymbol{\beta}$ to be strictly positive. This enforces that each tuple in the sample gets $w(t) \geq 0$ and is represented in the population.

Further, as we want to avoid $w(t) = 0$, we add an additional row of $[n_S, 0, \ldots, 0]$ with $\sum_{i=1}^{m} N_i$ zeros to the matrix $\boldsymbol{G}^{0/1}\boldsymbol{X}_S$ and add the associated value of n_S to $\boldsymbol{y}$. This encourages the intercept value to be positive, which will force every tuple to get some positive weight (since the $\boldsymbol{\beta}$ parameters are already positive). If a $w(t)$ does get set to 0, we set $w(t) = 1$.

Example 4.2. *Continuing with Example 3.1. The one-hot encoded version of S,* $\boldsymbol{X}_S$*, is*

$$\begin{matrix} 1s & d_{01} & d_{02} & o_{FL} & o_{NC} & o_{NY} & d_{FL} & d_{NC} & d_{NY} \end{matrix}$$
$$\begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}.$$

Our aggregate matrix is

$$\boldsymbol{G}^{0/1} = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{matrix} \texttt{date} = 01 \\ \texttt{date} = 02 \\ \texttt{o\_st} = FL \;\&\; \texttt{d\_st} = FL \\ \texttt{o\_st} = FL \;\&\; \texttt{d\_st} = NY \\ \texttt{o\_st} = NC \;\&\; \texttt{d\_st} = FL \\ \texttt{o\_st} = NC \;\&\; \texttt{d\_st} = NY \\ \texttt{o\_st} = NY \;\&\; \texttt{d\_st} = FL \\ \texttt{o\_st} = NY \;\&\; \texttt{d\_st} = NC \\ \texttt{o\_st} = NY \;\&\; \texttt{d\_st} = NY \end{matrix}$$

where the right-most column shows the group attributes corresponding to the row in the matrix. After $\boldsymbol{G}^{0/1}\boldsymbol{X}_S$ *is calculated, we add the row of* $[4, 0, \ldots, 0]$ *at the bottom. Finally, our solution vector is*

$$\boldsymbol{y} = \begin{bmatrix} 5 & 5 & 2 & 1 & 1 & 3 & 1 & 1 & 1 & 4 \end{bmatrix}^T$$

where the final 4 is from adding the n_S *constraint to* $\boldsymbol{y}$*.*

With these two changes, we solve for $\boldsymbol{\beta}$ and $w(t) = \boldsymbol{\beta} \cdot \boldsymbol{t}^{0/1}$. The final processing step is to modify $w(t)$ so that $\sum_{t \in S} w(t) = n$. This is a simple multiplicative update to each $w(t)$ so that $w(t) = \frac{n}{\sum_{t \in S} w(t)} w(t)$. We do this to sum-normalize the weights to correctly reflect to true size of the population *after* learning $w(t)$. Note that uniform reweighting is equivalent to setting $w(t) \equiv 1$ before sum-normalizing.

Algorithm 1 IPF [51]

```
iter ← 0
while not converged or iter < maxIter do
    for j = 1 to ∑_{i=1}^{B} M_i do
        if G^{0/1}[j] · w ≠ y[j] then
            s ← y[j] / (G^{0/1}[j] · w)
            for i = 1 to n_S do
                if G^{0/1}[j][i] = 1 then
                    w[i] ← s * w[i]
    iter ← iter + 1
end while
```

4.1.2 Iterative Proportional Fitting. An alternative approach to finding $w(t)$ is to assume every $w(t)$ is independent and can be solved for directly; i.e. $w(t)$ is not a function of its attributes. Inspired by the technique of population synthesis in demography, we apply a technique called Iterative Proportional Fitting (IPF) [11, 23, 27, 45, 51, 61] to solve for $w(t)$. While IPF is not new, our approach of using IPF for arbitrary data debiasing is novel.

To briefly review IPF, IPF is a simple iterative procedure for calibrating sample weights to match given population aggregates and is traditionally used to reweight representative microsamples of some population to aggregate census reports. For each individual aggregate, if that aggregate is not satisfied in the sample, the weights of the participating tuples are rescaled to satisfy the selected aggregate. The procedure continues to iterative over aggregates until all aggregates are satisfied. It converges to a satisfactory scaling if such a scaling exists[2]. If no scaling exists, the algorithm may not converge and can only give an approximate reweighting.

The iterative algorithm begins by building the same incidence matrix, $\boldsymbol{G}^{0/1}$, as before, where each row represents a single constraint and each column represents a tuple in S. $\boldsymbol{y}$ is the vector of all aggregate queries' count values. With IPF, however, we have no $\boldsymbol{X}_S$. Instead, we have a n_S sized vector $\boldsymbol{w}$ of the weights of each tuple; i.e. $\boldsymbol{G}^{0/1}\boldsymbol{w} = \boldsymbol{y}$. At each iteration, a value in $\boldsymbol{w}$ is updated so that its associated aggregate constraint is satisfied.

The pseudocode for IPF is shown in Alg. 1 where $[j]$ represents getting row j for matrices and element j for vectors. At each iteration, if the dot product of the jth row of $\boldsymbol{G}^{0/1}$ with $\boldsymbol{w}$ does not equal the jth element in $\boldsymbol{y}$, the weights are scaled so that the constraint is satisfied. Note that only the weights participating in the aggregate, i.e. with nonzero $\boldsymbol{G}^{0/1}[j]$ values, are updated.

While both the reweighting techniques solve for $w(t)$, that linear regression has $m^{0/1}$ parameters while IPF has n_S parameters. Typically, $m^{0/1} < n_S$. Further, since both methods have $\sum_{i=1,B} M_i$ constraints, typically, linear regression is over constrained while IPF is under constrained.

[2]IFP is the same algorithm as in matrix scaling, the RAS algorithm, and biproportional fitting [45, 59].

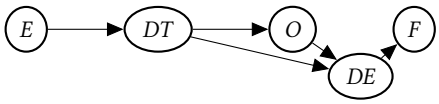

Figure 2: Example Bayesian network of flights in the United States (see Table 2 for abbreviations).

4.2 Probabilistic Model Learning

We just presented two different reweighting schemes to debias S using the aggregates Γ. It is important to understand when sample reweighting will fail. For one, the Horvitz-Thompson estimator, which we are approximating by $w(t)$, assumes the support of the sample is the same as the population, i.e. $Pr_S(t) > 0\ \forall t$. When this does not hold, e.g. when the sampling design is flawed, sample reweighting is inaccurate. Secondly, even if the support is the same, sample reweighting will fail when tuples exist in P but not in S because the sample will always say those tuples do not exist. This occurs with rare groups and small sample sizes. While we could impute missing rows to S, this risks losing important structural information (S gives us partial information about the manifold P lives on) and slowing down queries.

This section presents our solution to this problem: build a probabilistic model of P using S and Γ and answer queries directly over this model [24, 56]. In order to reason about the population probability distribution, we use the possible world semantics. As existing model learning techniques assume access to P, we present unique modifications and enhancements to adapt these techniques to use Γ and S.

When building a probabilistic model, the first consideration is what class of distributions to use. For example, if the population is belived to be Gaussian in nature, learning a mixture of Gaussians will likely be optimal. As we have no prior knowledge on the population, our main concern is choosing a distribution that can be learned from aggregate data. Similar to [61], we use a Bayesian network (BN) to model the population distribution as a Bayesian network is parameterized by aggregate queries and can scale to many attributes and large data [29]. Unlike [61], which builds the BN from the sample only, the novelty of our BN framework is that is merges S and Γ into BN learning.

4.2.1 Bayesian Networks Overview. We will briefly review Bayesian networks before discussing our framework. A Bayesian network is a probabilistic graphical model representing a set of random variables and their conditional dependencies through a directed, acyclic graph. Each edge represents a conditional dependency

of the form $\Pr(X_i|Pa(X_i))$ where $Pa(X_i)$ are the parents of node X_i. The example in Fig. 2 represents a Bayesian network modeling flights in the United States[3]. The joint distribution is $\Pr(E, DT, O, DE, F) = \Pr(DT|E)\Pr(O|DT)\Pr(DE|O, DT)\Pr(F|DE)\Pr(E)$.

Assuming access to P, there are numerous techniques for learning the BN's structure [14, 29, 47]. Once the Bayesian network structure is known, the parameters $\theta_{i,j,k}$, where $\Pr(X_i = j|Pa(X_i) = k) = \theta_{i,j,k}$, are calculated by minimizing the negative log likelihood of the joint probability subject to constraints enforcing each factor is a (conditional) probability; i.e. for some $Pa(X_i) = k$, $\sum_j \Pr(X_i = j|Pa(X_i) = k) = 1$ and $\Pr(X_i = j|Pa(X_i) = k) \geq 0$. With full access to P, a closed from solution exists that sets the parameters to be the fraction of times each possible state of X_i occurs given the possible states of $Pa(X_i)$ (an aggregate query).

Our challenge is how to do structure learning and parameter learning *without* access to P.

4.2.2 Learning Network Structure. To exclusively use S and Γ to learn the structure, we adapt the greedy hill-climbing algorithm [30, 47]. The traditional hill-climbing algorithm's goal is to find the structure that maximizes some score. At each step of the algorithm, it makes the "move" the improves the score the most. A "move" is either adding, removing, or reversing a directed edge. If the score cannot be further improved, the algorithm terminates.

We modify the algorithm as follows. To focus on learning from the population before the sample, our algorithm runs in two phases: building from Γ and building from S. As Γ represents ground truth information, we want to build as many edges from Γ before adding edges from S. In the first phase, we make "moves" using Γ until all attributes from Γ are added to the network. Then, if there are any remaining attributes in S not in Γ, we use S to continue building.

In the Γ building phase, we further modify the move selection algorithm to ensure we can score a candidate edge from X_i to X_j. As scoring requires computing a group by query over X_i, X_j, and $Pa(X_i)$, we only consider candidate edges that have the necessary support in Γ; i.e. the attributes X_i, X_j, and $Pa(X_i)$ appear together in some aggregate. In the S building phase, all edges are allowed.

Our last modification is to "lock in" edges that are added from the Γ phase, meaning we cannot remove them, because we want to keep all structural knowledge from Γ intact. This also prevents overfitting to the sample.

The pseudocode for our greedy hill-climbing algorithm is shown in Alg. 2. We use the BIC score because it discourages overly complicated structures that could overfit and does not depend on any prior over the parameters [47]. $\mathcal{E}$ is the set of directed edges, $\mathcal{T}$ is the set of conditional probability tables needed to parameterize the network, $\mathcal{P} \in \{1, 2\}$ is the phase of the algorithm, and s represents the BIC score. The functions `CondProbTables` and `BIC` are the same as in the standard algorithm. The function `BuildEdges`, shown in Alg. 3, determines which moves are allowed; i.e., if an edge has the necessary support.

[3] F (flights date), O (origin), DE (destination), E (elapsed time), DT (distance)

Algorithm 2 GreedyHC

$\mathcal{E} \leftarrow \emptyset$, $\mathcal{N} \leftarrow$ all nodes
$s \leftarrow -\infty$, $s' \leftarrow -\infty$, $\mathcal{P} = 1$
do
  if $\mathcal{P} = 1$ **then** $\mathcal{D} \leftarrow \Gamma$
  else $\mathcal{D} \leftarrow S$
  for $(X_i, X_j) \in \mathcal{N} \times \mathcal{N}$ **do**
    for $\mathcal{E}' \in$ BuildEdges$((X_i, X_j), \mathcal{E}, D, \mathcal{P})$ **do**
      $\mathcal{T}' \leftarrow$ CondProbTables$(\mathcal{D}, \mathcal{E}')$
      $t \leftarrow \text{BIC}(\mathcal{T}', \mathcal{E}')$
      if $t > s'$ **then** $s' \leftarrow t$
  if $attrs(\Gamma) \in \mathcal{E}$ and $|s - s'| \leq 0$ **then** $\mathcal{P} \leftarrow 2$
while $|s - s'| > 0$
return $\mathcal{E}, \mathcal{T}$

Algorithm 3 BuildEdges

if $\mathcal{P} = 2$ and (X_i, X_j) added when $\mathcal{P} = 1$ **then**
  $\mathcal{S} \leftarrow \{\mathcal{E} - (X_i, X_j)\}$
if $\{X_j, X_i, Pa(X_j)\} \in attrs(\mathcal{D})$ **then**
  $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{E} \cup (X_i, X_j)$
if $\{X_j, X_i, Pa(X_i)\} \in attrs(\mathcal{D})$ **then**
  $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{E} - (X_i, X_j) \cup (X_j, X_i)$
return $\mathcal{S}$

4.2.3 Learning Network Parameters. We must adapt the standard parameter learning optimization framework to use Γ and S. Inspired by [22, 55] adding parameter sharing and value equality constraints, we add constraints enforcing each aggregate is satisfied. Specifically, let $\mathcal{J}_i = \{j_{i,1}, \ldots, j_{i,d_i}\}$ be the attribute index set of aggregate i; i.e. $\boldsymbol{a}_{i,k} = [a_{k,j_{i,1}}, \ldots, a_{k,j_{i,d_i}}]$. Then, for some $(\boldsymbol{a}_{i',k'}, c_{i',k'}) \in \Gamma_{i'}$ (we use i' and k' to differentiate from the Bayesian network i and k variables), we add the constraint

$$\sum_{\boldsymbol{v} \in \bigtimes_{j' \in \neg\mathcal{J}_{i'}} dom(A_{j'})} \Pr(X_{\mathcal{J}_{i'}} = \boldsymbol{a}_{i',k'}, X_{\neg\mathcal{J}_{i'}} = \boldsymbol{v}) = \frac{c_{i',k'}}{n}$$

where dom is the active domain of an attribute, $\bigtimes$ is the cross product, $\neg\mathcal{J}_{i'} = \{1, \ldots, m\} - \mathcal{J}_{i'}$, $X_{\mathcal{J}_{i'}} = \boldsymbol{a}_{i',k'}$ stands for $X_{j_{i',1}} = a_{k',j_{i',1}}, \cdots, X_{j_{i',d_{i'}}} = a_{k',j_{i',d_{i'}}}$, and similarly for $X_{\neg\mathcal{J}_{i'}} = \boldsymbol{v}$. Intuitively, we are summing over all possible values of the attributes, $X_{\neg\mathcal{J}_{i'}}$, that do *not* participate in the aggregate.

Following Fig. 2, suppose we know that one aggregate attribute-value pair is that 0.2 percent of flights have $O =$ `KA`,

DE = NM, and ET = 60. The added constraint from that aggregate is

$$\sum_{dt \in dom(DT)} \sum_{f \in dom(F)} \theta_{DT,dt,\{60\}} * \theta_{O,\mathrm{KA},\{dt\}} * \theta_{DE,\mathrm{NM},\{\mathrm{KA},dt\}} * \theta_{F,f,\{\mathrm{NM}\}} * \theta_{E,60,\emptyset} = 0.2$$

We now get the constrained optimization problem in Eq. 2. $\theta_{i,j,k}$ for a particular i and tuple t means $j = t.A_i$ and $k = \{t.A_{i'} : X_{i'} \in Pa(X_i)\}$. $\boldsymbol{v}$ is the same as before in the sum over all possible values of the attributes that do not participate in the aggregates. $\boldsymbol{v}_{j'}$ stands for the individual value of attribute $A_{j'}$. The $*$ stands for the value of the parent, and it will be set to a value in $\boldsymbol{a}_{i',k'}$ or $\boldsymbol{v}$, depending on if it participates in the aggregate or not.

$$\text{minimize: } -\sum_{t \in S} \sum_{i=1}^{m} \log \theta_{i,j,k} \tag{2}$$

$$\text{subject to: } \theta_{i,j,k} \geq 0 \qquad \forall i,j,k$$

$$\sum_{j} \theta_{i,j,k} = 1 \qquad \forall i,k$$

$$\sum_{\boldsymbol{v}} \prod_{j' \in \neg \mathcal{J}_{i'}} \theta_{A_{j'},\boldsymbol{v}_{j'},*} \prod_{j \in \mathcal{J}_{i'}} \theta_{A_j,a_{k',j},*} = \frac{c_{i',k'}}{n} \qquad \forall (\boldsymbol{a}_{i',k'}, c_{i',k'})$$

4.2.4 Query Answering. Once we have learned the probability distribution of our population, we can answer selection (point) queries probabilistically by calculating $n * \Pr(X_1 = x_1, \ldots, X_m = x_m)$. To answer non-selection queries, such as top-k or group by, we use the BN to generate a sample of data that is representative of the population via forward/-logic sampling [35, 40, 61]. Once the sample S' is generated (this is not the original sample S), tuples are uniformly scaled up (i.e. the weight of each tuple is $|P|/|S'|$), and queries are answered as they are for reweighted samples.

4.3 Hybrid Query Evaluator

Our hybrid approach integrates the previous two methods into a unified technique for query answering. A naïve approach is to use the BN to reweight the sample. However, this approach does not solve the fact that tuples in the population do not exist in the sample. Our hybrid approach needs to rely on the sample only when a tuple exists in the sample.

Our hybrid approach is straightforward. When a point query gets issued, if the tuple being queried is in the sample, it uses the reweighted sample. Otherwise, it uses the BN. If the user issues a top-k query, it gets sent to the reweighted sample as the sample is more likely to contain heavy hitters, i.e. top-k values. In both cases, if the support of the sample is known to be different from the support of the population, the BN should always be used. It is future work to detect if the support is sufficient to use the reweighted sample.

The motivation of this demarcation between using the sample versus the Bayesian network is due to the inherent problems with sample reweighting. If the tuple does not exist or the support is not sufficient, the sample achieves poor accuracy. We are simply capturing this failure in our query evaluator by only using the sample when we believe it will achieve the lowest error.

5 OPTIMIZATION

We showed how to incorporate aggregates into sample reweighting and BN learning. The main challenges to efficient implementation are that every individual aggregate $(\boldsymbol{a}_{i,k}, c_{i,k})$ adds complexity to the optimization, e.g. each new aggregate is one more iteration of weight rescaling in IPF, and more importantly, the constrained optimization in Eq. 2 is a nonlinear optimization with $O(\prod_{j \in \neg \mathcal{J}_i} N_j)$ variables for *each* $(\boldsymbol{a}_{i,k}, c_{i,k})$. The large number of variables along with the added complexity of using nonlinear constraints [60] makes solving computationally expensive. Further, our constraints prevent us from optimizing each BN factor independently, a benefit of using traditional BNs.

To solve these problems, we present two optimization techniques: pruning the least informative aggregates and simplifying our constraint optimization.

5.1 Aggregate Selection

Our goal is to reduce the number of aggregates, i.e. $|\Gamma|$, *before* using them in reweighting and Bayesian network learning. The natural choice is to choose the B most informative aggregates; i.e., the B aggregates that minimize the distance between the true distribution and some distribution parametrized by the aggregates (more on this later). We chose to minimize the Kullback-Leibler (KL) divergence, which measures the amount of information lost between two distributions, because of the existing work on minimizing the KL divergence (e.g. Chow-Liu trees [19] and VAEs [25]).

To minimize the KL divergence, we assume, like BNs, that our approximate distribution is a product distribution because there is a known product distribution that minimizes the KL divergence from the true distribution: a Chow-Liu tree [19] (a second-order product approximate) and its higher-order extension, a k-order t-cherry junction tree [13, 62]. Although Γ can contain aggregates with $d_i > 2$, in Sec. 6, we limit ourselves to $d_i \leq 2$; therefore, we only present our optimization in terms of Chow-Liu trees (see [5] for extension to $d_i > 2$).

Recall that a Chow-Liu tree finds a tree structure the minimizes the KL divergence from the true distribution where each node represents an attribute. The algorithm to construct a Chow-Liu tree greedily adds undirected edges between nodes in descending order of the mutual information score

as long as the two nodes are disconnected. It terminates once a spanning tree is built.

We modify the Chow-Liu algorithm is two ways. First, as our algorithm does not have access to P, we only add edges that have support in Γ, meaning we can calculate the mutual information from Γ alone. Second, as our aggregate budget B may larger than the number of attributes, once the algorithm terminates, we restart it, building a new tree from scratch. To avoid creating duplicate tree edges, we disallow previous edges from being added. Once our algorithm generates B edges, we filter Γ so that each $\boldsymbol{\gamma}_i$ must be equal to the attributes associated with one of the edges.

5.2 Bayesian Network Simplification

To optimize our constrained optimization, we want each BN factor $\Pr(X_i|Pa(X_i))$ to be optimized independently with *linear* constraints. To do this, we enforce a topological solving order and limit the aggregates added to our model.

To understand how do this, we first need to discuss what a topological solving order is. For now, assume we can solve each BN factor independently. A topological solving order is simply solving for the factors in topological order, meaning every parent node is optimized before its child node. For example, if $X_{i'}$ is a parent of X_i, then we solve for $\theta_{i',j,k}$ before solving for $\theta_{i,j,k}$. Once $\theta_{i',j,k}$ is solved, denote it $\overline{\theta}_{i',j,k}$.

To enforce linear constraints and independent solving, we restrict our model to only add aggregate constraints that act on single factors, i.e. aggregate constraints over a child node X_i and its parents if the parents are in the aggregates [4]. This means for child node X_i, the constraints in Eq. 2 will only contain the the product of the child parameter $\theta_{i,j,k}$ with its ancestors because the other factors have marginalized out. By itself, this has only reduced the number of product factors. The key is topological solving order. By insuring that the parents are solved for before the children, at the time of solving for $\theta_{i,j,k}$ for a particular X_i, the ancestor terms are already known and become a constant in the constraint, meaning the $\theta_{i,j,k}$ for X_i are the only parameters. By removing aggregate constraints that act on multiple different BN factors, we can turn our nonlinear constraints into linear ones.

This linearity of constraints for a particular factor allows us solve for factors independently. As we only include constraints on single factors and only those factor's parameters are unknown, we can solve factors independently.

Example 5.1. *Take the example network from Fig. 2. Suppose we have two aggregates over E and* (O, DE)*. A topological ordering of all nodes is E, DT, O, DE, F. To simplify indexing, instead of using* i', k' *to index the aggregates, we will use the associated values as indexes for the counts. For example, if one aggregate of* (O, DE) *is* $(\boldsymbol{a}_{2,k'}, c_{2,k'}) = ([\texttt{WI}, \texttt{MN}], 10)$*, we write this as* $c_{2,\{\texttt{WI},\texttt{MN}\}} = 10$ *(2 representing the second aggregate).*[5]

We solve for E first by solving

$$\begin{aligned} \textit{minimize:} \quad & -\sum_{t \in S} \log \theta_{E,j,\emptyset} & \\ \textit{subject to:} \quad & \theta_{E,j,\emptyset} \geq 0 & \forall j \\ & \sum_{j} \theta_{E,j,\emptyset} = 1 & \\ & \theta_{E,j,\emptyset} = c_{1,j}/n & \forall j \in dom(E). \end{aligned}$$

Note that because of marginalization of the BN factors, the aggregate constraints do not sum over all possible values of (DT, O, DE, F)*. Once this optimization is solved, we have* $\overline{\theta}_{E,j,\emptyset}$*. We solve our next node, DT, in closed form because we have no constraints over DT. We solve O next by*

$$\begin{aligned} \textit{minimize:} \quad & -\sum_{t \in S} \log \theta_{O,j,k} & \\ \textit{subject to:} \quad & \theta_{O,j,k} \geq 0 & \forall j,k \\ & \sum_{j} \theta_{O,j,k} = 1 & \forall k \\ & \sum_{k \in dom(DT)} \theta_{O,j,k} \sum_{\ell \in dom(E)} \overline{\theta}_{E,\ell,\emptyset} \overline{\theta}_{DT,k,\ell} & \\ & \qquad = \sum_{v \in dom(DE)} \frac{c_{2,\{j,v\}}}{n} & \forall j \in dom(O). \end{aligned}$$

Note we turn the constraint over (O, DE) *to be one just over O by aggregation. We use* (O, DE) *again when solving for DE. F is solved in closed form.*

We can further improve efficiency by limiting the number of parents nodes each child can have in Bayesian network structure learning by modifying the hill-climbing algorithm to prevent adding/reversing edges if a child already has enough parents.

6 EVALUATION

In this section, we evaluate the query accuracy and query execution time of Themis. We compare Themis's hybrid approach to always using the optimal sample reweighting technique and always using the best Bayesian network technique. We further investigate the performance of the two different sample reweighting techniques (linear regression and IPF) and of Bayesian network learning technique variations. Lastly, we demonstrate the benefits of using our pruning technique. We do not plot results for our constraint solver

[4] If we have an aggregate over X_i and some other $X_{i'}$, we can turn that aggregate into one over just X_i by grouping by X_i, $Pa(X_i)$ and summing the counts.

[5] Even though both values are US States, the set construction is valid because they are really origin state WI and destination state MN.

Flights	Abrv
fl_date	F
origin_state	O
dest_state	DE
elapsed_time	E
distance	DT

IMDB	Abrv
movie_year	MY
movie_country	MC
name	N
gender	G
actor_birth	B
rating	RG
top_250_rank	TR
runtime	RT

Table 2: `Flights` and `IMDB` attributes.

optimization from Sec. 5.2 as experiments did not finish in under 10 hours without using the optimization.

6.1 Implementation

We implemented Themis in three parts: the sample reweighter, the BN learner, and the query evaluator (code at [6]). The linear regression, IPF, and BN constraint solving were implemented in Python 3.7[6]. The BN structure learning and inference were implemented in R 3.2 using the BNLearn and gRain package (gRain for exact inference). Lastly, we limited our Bayesian networks to be trees (i.e. each child can have at most one parent) to limit the number of tuning parameters to evaluate. For extended results, see [5].

After learning, the samples with weights stored as an additional column were stored and queried in a Postgres 9.5 database. We performed all experiments on a 64bit Linux machine running Ubuntu 16.04.5. The machine has 120 CPUs and 1 TB of memory. The Postgres database also resides on this machine with a shared buffer size of 250 GB.

6.2 Datasets

We use a flights dataset [7] (all United States flights in 2005 with $n = 6,992,839$), an IMDB dataset [42] (actor-movie pairs released in the United States, Great Britain, and Canada with $n = 846,380$), and a synthetic CHILD Bayesian network dataset generated using BNLearn [8] with $n = 20,000$. We preprocess the datasets to remove null values and bucketize the real-valued attributes into equi-width buckets. For the two real-world datasets, the attributes and attribute abbreviates are shown in Table 2.

We take three samples from `Flights`: uniform (Unif), flight month of June (June), and flights leaving from a four corner state of CA, NY, FL, WA (SCorners)[7]. Each are 10 percent samples with a 90 percent bias, meaning 90 percent of the rows are from the selection criteria. We also take a corner states 10 percent sample with 100 percent bias (Corners).

[6]Since the constraint solving was approximate, occasionally a network parameter, i.e. $\theta_{i,j,k}$, would be set to some very small negative number. We set these parameters to zero.

[7]The S stands for the supported Corners sample.

B	1	2	3	4
Flights	E & DT	DE & DT	O & DT	F & DE
IMDB	MY & RT	RG & RT	MY & MC	MY & G

Table 3: The 4 2D `Flights` and 4 2D `IMDB` data aggregate attributes chosen by the pruning technique.

We likewise take three samples from `IMDB`: uniform (Unif), movie country of Great Britain (GB), and movies with ratings 1, 5, or 9 (SR159). We similarly give these 10 percent samples a 90 percent bias and take a 10 percent sample with 100 percent bias of the ratings sample (R159).

As the CHILD data is used to examine our pruning technique, we just use a 10 percent uniform sample.

6.3 Experimental Setup

As real population reports typically have aggregates of one or two dimensions (e.g. Excel tables), we use $d = 1$ or 2. We prune all possible aggregates by our pruning technique to produce from $B = 1$ to 4 aggregates. Table 3 shows the aggregates chosen. For `IMDB`, we only consider aggregates from the attributes `MY`, `MC`, `G`, `RG`, `RT` to investigate the impact of having aggregates that do no cover all attributes.

To measure accuracy, we run top-k queries for $K = 30$ and point queries where the point query selection values are selected from the population's light hitters (smallest values), heavy hitters, and randomly. We run 100 point queries for each of the three selections, for a total of 300 point queries per attribute set.

For `Flights`, we issue point queries over all possible attribute sets of size two to five (total of 26). We run top-k queries for all possible two and three attribute set combinations (total of 20). For `IMDB`, as there are too many attributes to run all possible point queries, we randomly choose 20 three dimensional attribute sets to run our point and top-k queries over. Note we look at all attributes for queries, not just those we have aggregates over.

Following [67], we measure precision and recall of the top-k queries and report the F measure. For point queries, we use the error metric of percent difference, $2 * |true\_value - est\_value| / |true\_value + est\_value|$. We use percent difference rather than percent error to avoid over emphasizing errors where the true value is small and to ensure missed groups get the maximum error of 200 percent.

Lastly, when measuring runtime (Sec. 6.9), as all reweighted samples are stored and accessed the same, we only look at the runtime for one reweighted sample.

6.4 Overall Accuracy

We first evaluate which method is optimal for answering queries over different biased samples. Using $B = 4$, we

Hitters	Percentile	Unif	June	SCorners	Corners
Heavy	25	4.2	13.6	168.3	6.1
	50	1.8	69.7	61.9	2.7
	75	1.4	29.6	34.4	2.2
Light	25	∞	∞	∞	45
	50	1.7	1.7	1.7	1.4
	75	1.0	1.0	1.0	1.0

Table 4: Percent improvement of percentiles for hybrid compared to Unif for the queries from Fig. 3. The infinite value represents that hybrid has zero error.

compare Themis's hybrid approach (pink) to the best linear reweighting technique of IPF (orange) and to the best Bayesian network technique of BB (blue) (BB means it uses both Γ and S to learn the BN). We further compare against standard uniform reweighting as a baseline (red).

Fig. 3 and Fig. 4 show boxplots of the percent difference of 100 heavy and 100 light hitter point queries across the samples. The median value is the black line and the average is the black X. For reference, Table 4 shows the percent improvement of the 25th, 50th, and 75th percentiles of Themis's hybrid approach to uniform reweighting for `Flights`.

We see that for the samples that have the same support as the population (first three), Themis's hybrid technique achieves the lowest error. For the `Flights` sample without support (Corners), the BN technique (BB) performs best, but hybrid performs better that IPF, indicating that hybrid mitigates the problem of mismatching support. BB does not perform optimally for R159 because of queries over the very dense attribute `N` (48,000 distinct values). BB learns that `N` is uniformly distributed and underestimates queries over `N` because all values are equally likely. This makes BB perform worse than IPF. Further, as three attributes are not covered by aggregates, one of them (top 250 rank/`TR`) being highly correlated with the attribute biasing the sample, BB and IPF overfit to the sample and receive high error for queries over `TR`.

As the remaining trends we present are the same for `Flights` and `IMDB`, we only show `Flights` plots (see [5] for `IMDB` results). Fig. 5 shows the error of 100 random point queries for `Flights` grouped by if the query was in the sample or not. The separation mimics Themis's hybrid query evaluator decision, which is why hybrid receives the same error as IPF for queries in the sample and as BB for queries not in the sample. We see the same trend that IPF only performs well when the support is the same.

We now evaluate which method is optimal for answering top-k queries using `Flights` (trends are similar for `IMDB` except, due to the larger number of attributes compared to sample size, all methods score on average 17 percent lower on F measure). We dive more deeply into top-k performance by varying the number of 2D aggregates after adding 5 1D

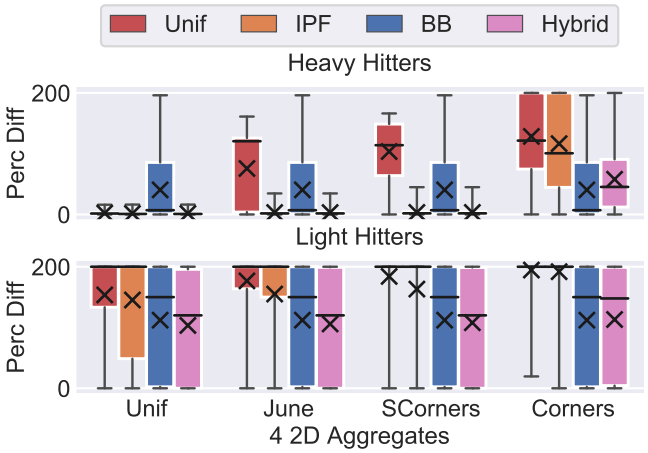

Figure 3: 100 heavy and light hitter point query percent difference for `Flights` biased samples ($B = 4$).

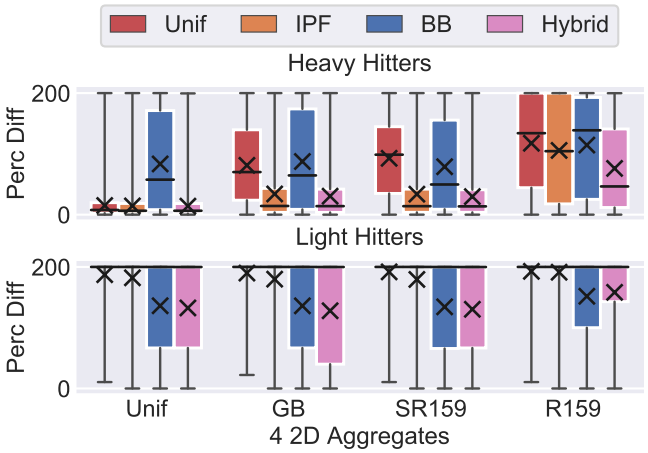

Figure 4: 100 heavy and light hitter point query percent difference for `IMDB` biased samples ($B = 4$).

aggregates, one for each attribute. We do not show results for other K values as the trends are similar.

Fig. 6 shows a dramatic improvement in the IPF/hybrid approach (they are the same for top-k) for SCorners compared to Corners, which supports the results shown for point queries. BB improves the most from adding more 2D aggregates as it is learning more structural and parameter information from the population.

Overall, we see that Themis's hybrid technique outperforms the alternatives for point queries and top-k queries and mitigates the problem of the sample and population having different support.

6.5 Changing Aggregate Knowledge

To examine how varying our aggregates impacts accuracy, we run 100 random point queries on two `Flights` samples as we add 1D and 2D aggregates. As trends are similar in

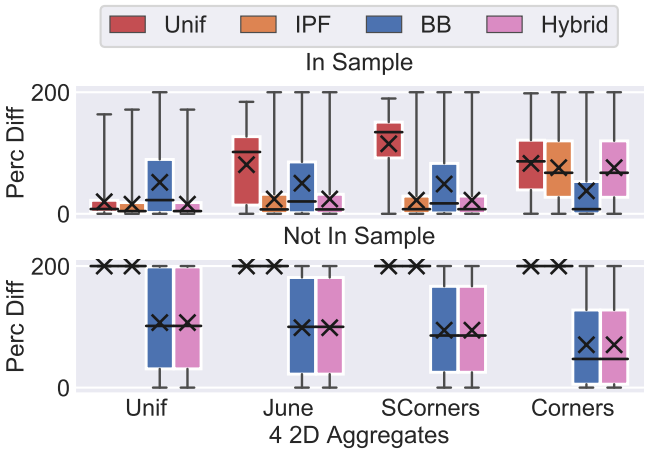

Figure 5: 100 random point query percent difference for `Flights` biased samples ($B = 4$).

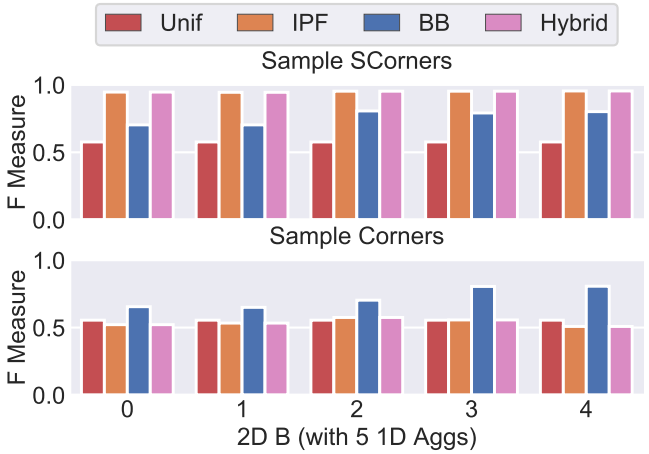

Figure 6: F measure of top 30 for 5 1D aggregates and varying numbers of 2D aggregates for `Flights`.

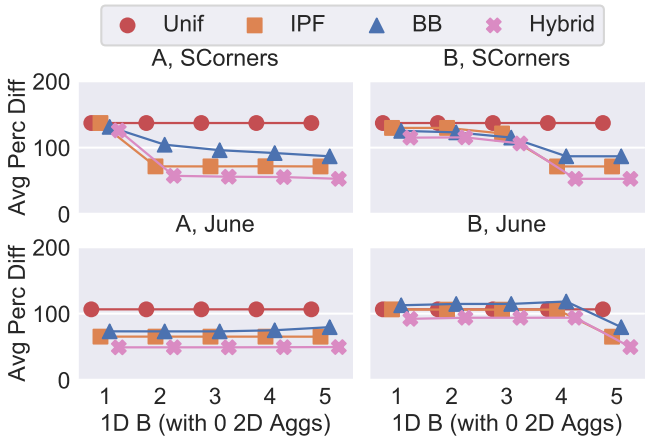

Figure 7: Average percent difference of 100 random point queries for SCorners and June for `Flights` as more 1D aggregates are added.

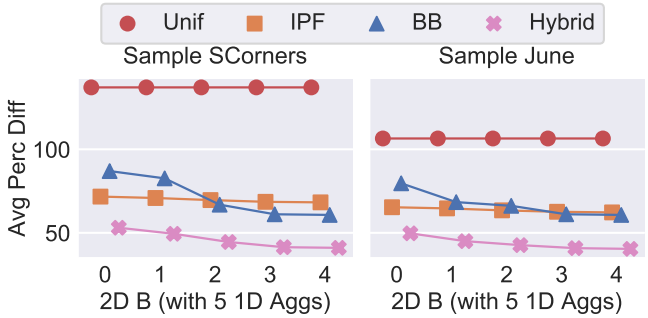

Figure 8: Average percent difference of 100 random point queries for SCorners and June for `Flights` as more 2D aggregates are added after adding the 5 1D aggregates from Fig. 7.

IMDB, we do not show those plots (see [5] for results). To show the impact of attribute coverage, we add the 1D aggregates in two different orders. For Flights, we add the 1D aggregates in order A—F, O, DE, E, DT—and order B, the reverse of order A. After adding the 1D aggregates, the 2D aggregates are added as in Table 3.

Fig. 7 shows a line plot (to better show trends) of the average percent difference for SCorners (top row) and June (bottom row) for order A (left column) and order B (right column). For SCorners, the largest improvement in all Themis methods (IPF, BB, and hybrid) is when adding the second attribute in order A (fourth attribute in order B). This is O (the attribute SCorners is biased on) which indicates that Themis is correctly learning which attribute is causing bias. The result is replicated with the June sample and attribute F.

Fig. 8 shows the average percent difference for the same two samples as 2D aggregates are added. We see that BB improves the most with more aggregates. However, we see diminishing returns after adding 2 aggregates. Once 4 2D aggregates are included, we see that the error for IPF, BB, and hybrid are approximately the same, indicating that when enough population data is added, knowledge from the sample is overwritten by the population data.

Lastly, to examine how the amount of sample bias impacts accuracy, we measure the average percent difference for 100 random point queries using 4 2D aggregates on the Flights sample of Corners as we decrease the percent bias from 100 percent (Corners sample) to 90 percent (SCorners sample), shown in Fig. 9. As soon as the support is the same (bias < 100), sample reweighting techniques start performing significantly better. Themis's hybrid approach is able to mitigate this difference and performs better than IPF for 100 percent bias.

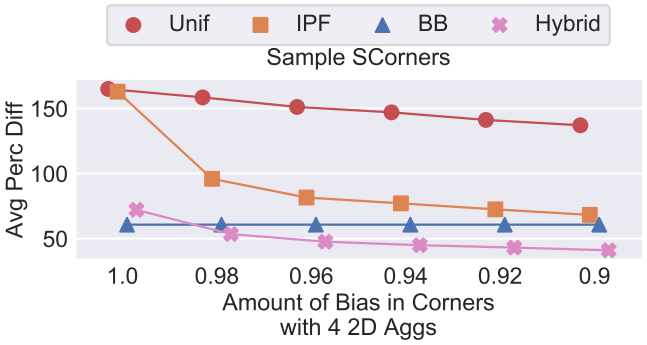

Figure 9: Average percent difference of 100 random point queries for the SCorners using 4 2D aggregates as we decrease the percent bias.

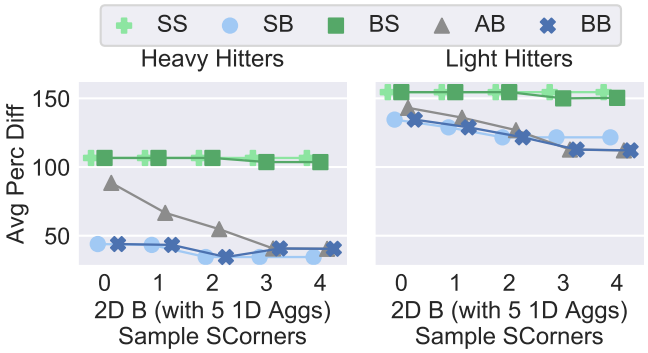

Figure 10: Average percent difference of 100 heavy hitter and light hitter queries over SCorners for the 5 different Bayesian techniques as more 2D aggregates are added with 5 1D aggregates already included.

6.6 Bayesian Network Performance

We now dive into the different BN approaches for structure and parameter learning. When learning a BN, we can use only the sample (S) or both the sample and aggregates (B). Since the aggregate information (A) is not always covering, we cannot use it exclusively to learn the structure (unless making suboptimal uniformity assumptions) or the parameters. Therefore, the approaches we compare are SS (light green), BS (dark green), SB (light blue), and BB (dark blue) where the letters represent whether S or B is used to learn the structure (first) and parameters (second). In addition, we compare against AB (grey) which is when the structure is learned just from the Γ. For attributes not covered by Γ, we add them as disconnected nodes (uniformity assumption). We measure average percent difference on 100 heavy and light hitter point queries on SCorners while increasing the number of 2D aggregates after adding 5 1D aggregates.

Fig. 10 shows average percent difference for heavy and light hitters for SCorners. We see that all approaches perform better for heavy hitters than light hitters and that BB performs best overall. As the number of aggregates increases, AB converges to BB because the population information overrides any sample information given for `Flights`.

Another important trend is that using the sample versus both the aggregates and the sample is more important for parameter learning than for structure learning. We see that while BS is slightly more accurate than SS, SB performs better than both.

6.7 Sample Reweighting Performance

We now compare the two different sample reweighting techniques of linear regression (LinReg) and IPF. As we do not see a drastic improvement in adding 2D aggregates when doing sample reweighting, we focus on comparing how they perform on the different `Flights` samples by measuring error on 100 random point queries using 4 2D aggregates.

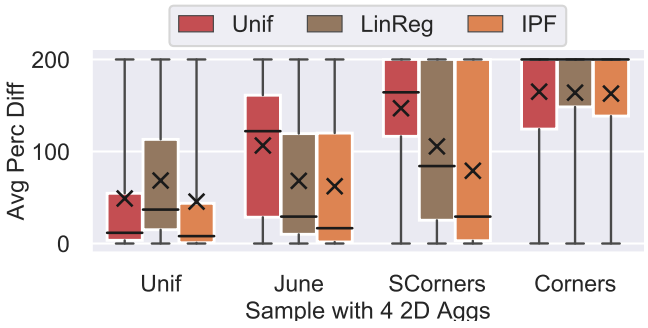

Figure 11: Percent difference of 100 random point queries over four different `Flights` samples with 4 2D aggregates.

Fig. 11 shows boxplots of the percent difference of LinReg, IPF, and Unif. We see clearly that IPF outperforms LinReg on all cases. While LinReg does outperform Unif in all biased samples (not the uniform sample), it does not outperform IPF due to the correlations that exist in the data. For example, to satisfy aggregates on the `DT` attribute, LinReg will add weight to the highly correlated attribute values of `E`. This will help satisfy aggregates on `DT` but will overall hurt performance because any other tuple with the correlated attribute values will have an incorrect weight.

6.8 Pruning Effectiveness

We use the BNLearn CHILD dataset to examine our pruning technique for choosing aggregates. We use the two Bayesian approaches of BB and AB and measure average percent difference for 100 random point queries for 10 randomly chosen attribute sets of size 2, 4, 6, 8, and 10 for a total of 50 attributes sets and 5,000 total point queries. We compare the techniques as we add from 5 to 65 2D aggregates using our pruning technique (Prune) and randomly (Rand) after adding full 1D aggregates. In addition to comparing the error, we

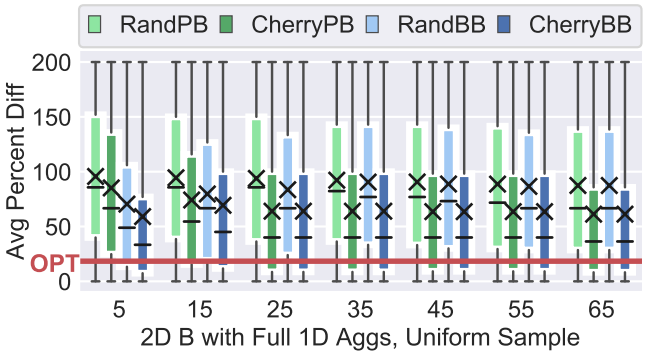

Figure 12: Average percent difference of 100 random point queries using a 10 percent uniform sample of the CHILD dataset with full 1D aggregates as more 2D aggregates are added using Prune and Rand.

Method	RW	SB	AB	SS	BS	BB
Runtime (10^{-3} s)	25.3	2.49	1.97	2.45	2.26	2.07

Table 5: Average query execution time of 100 random point queries over SR159 with 4 2D aggregates .

also plot the average percent difference if the true Bayesian network is known (optimal error).

Fig. 12 shows the average percent difference where the dark colors are for Prune while the light colors are for Rand. The red line shows the median optimal error. We see, again, that BB performs better the AB, especially when fewer aggregates are added. Further, the error with Rand improves more slowly than using Prune to add aggregates. If enough aggregates are added, however, the techniques eventually converge.

6.9 Execution Time

Lastly, we examine the query execution time and solving time of the different approaches. We run all timing experiments on the IMDB SR159 sample as IMDB has the larger active domain. For the query execution time, we run 100 random points queries.

Table 5 shoes the average query execution time of the six methods. As the query execution time does not noticeably change as we add more 2D aggregates, we only show results for 4 2D aggregates.

As the main bottle neck to using these methods is the solving time, we report the time it takes to learn the structure for BB (the solving times of the other Bayesian methods are comparable or faster) and the time it takes to learn the parameters of LinReg, IPF, and BB in Table 6 as we increase the number of aggregates.

We see that the structure learning time is negligible compared to the solver time for BB. LinReg is the fastest, followed by IPF, and then by BB. The solver time increases with all

	1D	1	2	3	4	5	5			
	2D	0					1	2	3	4
S	BB	0.3	0.3	0.4	0.7	0.7	2.3	5.3	8.4	13.2
P	Reg	0.4	0.5	0.6	1.0	3.1	4.6	6.1	6.6	7.0
	IPF	0.2	0.6	0.8	1.7	10.8	16.1	22.5	30.0	38.5
	BB	75.1	103	103	161	295	148	68.7	63.0	58.8

Table 6: Structure (S) and parameter (P) learning execution times in seconds for LinReg (denoted Reg for space), IPF, and BB using SR159 sample as 1D and 2D aggregates are added.

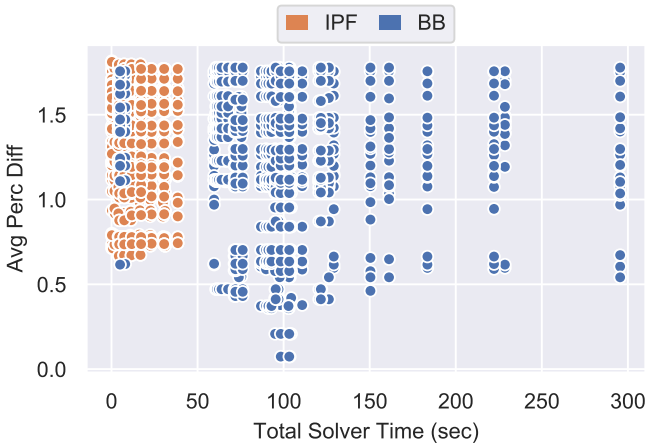

Figure 13: Average percent difference versus total solver time in seconds for IPF and BB on SR159 for various 1D and 2D aggregates.

methods as we increase the number of 1D aggregates. Surprisingly, as we add more 2D aggregates, the solving time of BB decreases. This is because as we add more 2D constraints to our model, the constraint solver has more direct equality constraints which are instantaneous to solve for.

To show this trend more directly, Fig. 13 shows the average percent difference for 100 random point queries compared to total solver time (structure learning plus parameter learning) for BB and IPF on SR159 while using different combinations of 1D and 2D aggregates. Specifically, we use our pruning technique to add one to four 2D aggregates that cover from one up to five attributes (the 5 attributes of MY, MC, G, RG, RT)[8]. We see that while IPF is almost always faster than BB to solve, BB is capable of achieving lower error. Further, there is an optimal spot around 100 seconds of solving time for BB which achieves the lowest error. This is when the most 2D aggregates are added to the model.

[8]If we use a numeric index from 1 to 5 into the 5 covered IMDB attributes, we choose different aggregates that cover the following sets of attributes: {{1},{2},{3},{4},{5},{1,2},{4,5},{1,2,3},{3,4,5},{1,2,3,4},{2,3,4,5},{1,2,3,4,5}}. For each attribute set, we run our pruning technique to choose one, two, three, and four 2D aggregates, if applicable (we can't add 4 unique 2D aggregates for only two attributes). If any of the attributes are not covered by a 2D aggregates, we add 1D aggregates so all attributes are covered.

7 RELATED WORK

Our technique is primarily related to population synthesis. Population synthesis' goal is to directly generate a population dataset from a sample and either historical population data [46] or aggregate data (typically geographical aggregates) using techniques such as IPF, Bayesian networks, constrained optimization, or MCMC and Gibb's sampling [11, 27, 41, 45, 51, 61]. Themis, however, combines two different techniques, does not assume the sample is representative of the population, and more accurately answers queries over tuples not in the sample.

Themis is similar to bootstrapping, which is a resampling technique for understanding sample bias for approximate query processing [17, 44, 50], but Themis does not suffer from the large overhead of doing bootstrapping and does not assume the sample is representative of the population.

The papers from Van den Broeck and Martin Grohe [15, 32], extend probabilistic databases to assume an open world and allow unknown values to exist. While Themis also assumes an open world, we are not a probabilistic database.

From a data integration standpoint, Themis is closely related to answering queries over views (samples) [33, 43] where the views do not contain complete information. However, we attempt to model the data missing from the data sources whereas data integration deals with knowing when answers are certain or not.

In regards to data cleaning [16, 38, 65], while Themis is trying to infer missing values from the sample, we are missing entire rows, not just attribute values.

There has also been a lot of research in the machine learning community in two related areas: one class classification [31, 39] and learning from aggregate labels [12, 18, 53]. The main difference is that Themis is trying to learn a classifier with aggregate data and does not have aggregate labels of both classes, i.e. in the sample and not in the sample.

Our method, in essence, calculates a propensity score for a record [49, 58]. However, standard propensity score techniques require data that is not in the sample to be given, which we do not have. A possible solution is to generate the data outside of the sample [34]. We leave this possible technique as future work.

The work of [20] is a particular subset of our problem. That work tries to take into account unknown unknown values when estimating aggregate queries. Our goal is similar, but we take a machine learning approach to re-weight samples rather than estimating missing values.

Themis is similar to [28, 70] which tries to remove bias from machine learning algorithms, i.e. make socially fair classifiers. Our research, however, does not have protected attributes nor does it have access to the entire population, but our methods could be used to help achieve fairness by, for example, forcing the aggregates over protected attributes to be uniform.

Considering selection bias and machine learning is the work of [36, 68]. Themis, however, only has access to the biased test set, i.e. sample, and does not have sample probabilities to use.

Although not concerned with debiasing data, the work of unioning tables [54] aims to find attributes among various tables that are unionable. The key similarity to Themis is that the tables are from different sources with different biases. Our research takes the output of [54], a set of unionable tables, and debiases them.

Also using aggregates as constraints is [63]. They use aggregates to constraint query answers; e.g., select tuples such that the average of the selection is below some value. The work in [48] discusses using Bayesian networks to approximate a data cube. Our work is similar to both of these except our goal is not to merely answer queries but to also debias a sample.

Similar to how Themis treats the biased samples as first class citizens, FactorBase [57] and BayesStore [64, 66] do the same with graphical models and probabilistic inference. While they both utilize Bayesian networks to model data, their goal is not data debiasing.

The work of [30] uses Bayesian networks over relations to do selectivity estimation for queries. Themis also uses Bayesian networks but is not concerned with multiple relations or selectivity estimation.

[52] performs conjunctive query selectivity estimation using both samples and synopses. Themis also uses samples and synopses where our synopses are of the form of population histograms. Our overall goal, however, is to debias the data. This is different than selectivity estimation, and unlike selectivity estimation, we do not control how we sample.

8 CONCLUSION

We present Themis, the first prototype open world database system that uses a biased sample and population-level aggregate information to answer queries approximately as if asked over the population. More importantly, our data debiasing is automatic. Themis's hybrid approach merges sample reweighting with population probabilistic modeling to achieve a 70 percent improvement in the median error when compared to uniform reweighting for heavy hitter queries. Further, as shown in Fig. 9, Themis is robust to differences in the support between the sample and population.

Future work is to automatically detect if support is sufficient to use the reweighted sample, integrate multiple samples into the debiasing process, and investigate alternative techniques to integrate the sample and population probabilistic model.

REFERENCES

[1] https://www.congress.gov/115/bills/s760/BILLS-115s760is.pdf.
[2] http://infusecp.mimas.ac.uk.
[3] https://pdf.ic3.gov/2017_IC3Report.pdf.
[4] https://dawn.cs.stanford.edu/2018/04/11/db-community/.
[5] anonymous for double blind.
[6] anonymous for double blind.
[7] https://www.transtats.bts.gov/.
[8] http://www.bnlearn.com/bnrepository/discrete-medium.html.
[9] P. C. Austin. An introduction to propensity score methods for reducing the effects of confounding in observational studies. *Multivariate behavioral research*, 46(3):399–424, 2011.
[10] J.-F. Beaumont. A new approach to weighting and inference in sample surveys. *Biometrika*, 95(3):539–553, 2008.
[11] R. J. Beckman, K. A. Baggerly, and M. D. McKay. Creating synthetic baseline populations. *Transportation Research Part A: Policy and Practice*, 30(6):415–429, 1996.
[12] A. Bhowmik, J. Ghosh, and O. Koyejo. Generalized linear models for aggregated data. In *Artificial Intelligence and Statistics*, pages 93–101, 2015.
[13] J. Bukszár and A. Prekopa. Probability bounds with cherry trees. *Mathematics of Operations Research*, 26(1):174–192, 2001.
[14] L. M. d. Campos. A scoring function for learning bayesian networks based on mutual information and conditional independence tests. *Journal of Machine Learning Research*, 7(Oct):2149–2187, 2006.
[15] I. I. Ceylan, A. Darwiche, and G. Van den Broeck. Open-world probabilistic databases. In *Description Logics*, 2016.
[16] S. Chaudhuri, A. Das Sarma, V. Ganti, and R. Kaushik. Leveraging aggregate constraints for deduplication. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 437–448. ACM, 2007.
[17] S. Chaudhuri, B. Ding, and S. Kandula. Approximate query processing: No silver bullet. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 511–519. ACM, 2017.
[18] B.-C. Chen, L. Chen, R. Ramakrishnan, and D. R. Musicant. Learning from aggregate views. In *Data Engineering, 2006. ICDE'06. Proceedings of the 22nd International Conference on*, pages 3–3. IEEE, 2006.
[19] C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE transactions on Information Theory*, 14(3):462–467, 1968.
[20] Y. Chung, M. L. Mortensen, C. Binnig, and T. Kraska. Estimating the impact of unknown unknowns on aggregate query results. *ACM Transactions on Database Systems (TODS)*, 43(1):3, 2018.
[21] C. Cortes, M. Mohri, M. Riley, and A. Rostamizadeh. Sample selection bias correction theory. In *International Conference on Algorithmic Learning Theory*, pages 38–53. Springer, 2008.
[22] C. P. De Campos, Y. Tong, and Q. Ji. Constrained maximum likelihood learning of bayesian networks for facial action recognition. In *European Conference on Computer Vision*, pages 168–181. Springer, 2008.
[23] W. E. Deming and F. F. Stephan. On a least squares adjustment of a sampled frequency table when the expected marginal totals are known. *The Annals of Mathematical Statistics*, 11(4):427–444, 1940.
[24] A. Deshpande and S. Madden. Mauvedb: supporting model-based user views in database systems. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 73–84. ACM, 2006.
[25] C. Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.
[26] C. Dwork. Differential privacy and the us census. In *Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 1–1. ACM, 2019.
[27] B. Farooq, M. Bierlaire, R. Hurtubia, and G. Flötteröd. Simulation based population synthesis. *Transportation Research Part B: Methodological*, 58:243–263, 2013.
[28] M. Feldman, S. A. Friedler, J. Moeller, C. Scheidegger, and S. Venkatasubramanian. Certifying and removing disparate impact. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 259–268. ACM, 2015.
[29] N. Friedman, I. Nachman, and D. Peér. Learning bayesian network structure from massive datasets: the "sparse candidate" algorithm. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 206–215. Morgan Kaufmann Publishers Inc., 1999.
[30] L. Getoor, B. Taskar, and D. Koller. Selectivity estimation using probabilistic models. In *ACM SIGMOD Record*, volume 30, pages 461–472. ACM, 2001.
[31] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.
[32] M. Grohe and P. Lindner. Probabilistic databases with an infinite open-world assumption. *arXiv preprint arXiv:1807.00607*, 2018.
[33] A. Y. Halevy. Answering queries using views: A survey. *The VLDB Journal*, 10(4):270–294, 2001.
[34] H. He and E. A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge & Data Engineering*, (9):1263–1284, 2008.
[35] M. Henrion. Propagating uncertainty in bayesian networks by probabilistic logic sampling. In *Machine Intelligence and Pattern Recognition*, volume 5, pages 149–163. Elsevier, 1988.
[36] J. Huang, A. Gretton, K. Borgwardt, B. Schölkopf, and A. J. Smola. Correcting sample selection bias by unlabeled data. In *Advances in neural information processing systems*, pages 601–608, 2007.
[37] M. Idel. A review of matrix scaling and sinkhorn's normal form for matrices and positive maps. *arXiv preprint arXiv:1609.06349*, 2016.
[38] I. F. Ilyas, X. Chu, et al. Trends in cleaning relational data: Consistency and deduplication. *Foundations and Trends® in Databases*, 5(4):281–393, 2015.
[39] S. S. Khan and M. G. Madden. A survey of recent trends in one class classification. In *Irish conference on artificial intelligence and cognitive science*, pages 188–197. Springer, 2009.
[40] K. B. Korb and A. E. Nicholson. *Bayesian artificial intelligence*. CRC press, 2010.
[41] D.-H. Lee and Y. Fu. Cross-entropy optimization model for population synthesis in activity-based microsimulation models. *Transportation Research Record*, 2255(1):20–27, 2011.
[42] V. Leis, A. Gubichev, A. Mirchev, P. Boncz, A. Kemper, and T. Neumann. How good are query optimizers, really? *Proceedings of the VLDB Endowment*, 9(3):204–215, 2015.
[43] M. Lenzerini. Data integration: A theoretical perspective. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 233–246. ACM, 2002.
[44] K. Li and G. Li. Approximate query processing: what is new and where to go? *Data Science and Engineering*, 3(4):379–397, 2018.
[45] R. Lovelace, M. Birkin, D. Ballas, and E. van Leeuwen. Evaluating the performance of iterative proportional fitting for spatial microsimulation: new tests for an established technique. *Journal of Artificial Societies and Social Simulation*, 18(2), 2015.
[46] M. C. Lovell. Seasonal adjustment of economic time series and multiple regression analysis. *Journal of the American Statistical Association*, 58(304):993–1010, 1963.
[47] D. Margaritis. Learning bayesian network model structure from data. Technical report, Carnegie-Mellon Univ Pittsburgh Pa School of Computer Science, 2003.

[48] D. Margaritis, C. Faloutsos, and S. Thrun. Netcube: A scalable tool for fast data mining and compression. In *VLDB*, pages 311–320, 2001.

[49] D. F. McCaffrey, B. A. Griffin, D. Almirall, M. E. Slaughter, R. Ramchand, and L. F. Burgette. A tutorial on propensity score estimation for multiple treatments using generalized boosted models. *Statistics in medicine*, 32(19):3388–3414, 2013.

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

[51] K. Müller. *A generalized approach to population synthesis.* PhD thesis, ETH Zurich, 2017.

[52] M. Müller, G. Moerkotte, and O. Kolb. Improved selectivity estimation by combining knowledge from sampling and synopses. *Proceedings of the VLDB Endowment*, 11(9):1016–1028, 2018.

[53] D. R. Musicant, J. M. Christensen, and J. F. Olson. Supervised learning by training on aggregate outputs. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pages 252–261. IEEE, 2007.

[54] F. Nargesian, E. Zhu, K. Q. Pu, and R. J. Miller. Table union search on open data. In *VLDB*, 2018.

[55] R. S. Niculescu. Exploiting parameter domain knowledge for learning in bayesian networks. *Technical Report CMU-TR-05-147*, 2005.

[56] L. Orr, M. Balazinska, and D. Suciu. Probabilistic database summarization for interactive data exploration. *Proceedings of the VLDB Endowment*, 10(10):1154–1165, 2017.

[57] Z. Qian and O. Schulte. Factorbase: Multi-relational model learning with sql all the way. In *Data Science and Advanced Analytics (DSAA), 2015. 36678 2015. IEEE International Conference on*, pages 1–10. IEEE, 2015.

[58] P. R. Rosenbaum and D. B. Rubin. Reducing bias in observational studies using subclassification on the propensity score. *Journal of the American statistical Association*, 79(387):516–524, 1984.

[59] R. Sinkhorn. Diagonal equivalence to matrices with prescribed row and column sums. *The American Mathematical Monthly*, 74(4):402–405, 1967.

[60] D. Solow. Linear and nonlinear programming. *Wiley Encyclopedia of Computer Science and Engineering*, 2007.

[61] L. Sun and A. Erath. A bayesian network approach for population synthesis. *Transportation Research Part C: Emerging Technologies*, 61:49–62, 2015.

[62] T. Szántai and E. Kovács. Discovering a junction tree behind a markov network by a greedy algorithm. *Optimization and Engineering*, 14(4):503–518, 2013.

[63] M. Vartak, V. Raghavan, E. A. Rundensteiner, and S. Madden. Refinement driven processing of aggregation constrained queries. In *EDBT*, pages 101–112, 2016.

[64] D. Z. Wang, E. Michelakis, M. Garofalakis, and J. M. Hellerstein. Bayesstore: managing large, uncertain data repositories with probabilistic graphical models. *Proceedings of the VLDB Endowment*, 1(1):340–351, 2008.

[65] J. Wang, S. Krishnan, M. J. Franklin, K. Goldberg, T. Kraska, and T. Milo. A sample-and-clean framework for fast and accurate query processing on dirty data. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 469–480. ACM, 2014.

[66] Z. Wang. *Extracting and Querying Probabilistic Information in BayesStore.* PhD thesis, UC Berkeley, 2011.

[67] R. C.-W. Wong and A. W.-C. Fu. Mining top-k frequent itemsets from data streams. *Data Mining and Knowledge Discovery*, 13(2):193–217, 2006.

[68] B. Zadrozny. Learning and evaluating classifiers under sample selection bias. In *Proceedings of the twenty-first international conference on Machine learning*, page 114. ACM, 2004.

[69] E. Zagheni, V. R. K. Garimella, I. Weber, et al. Inferring international and internal migration patterns from twitter data. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 439–444. ACM, 2014.

[70] R. Zemel, Y. Wu, K. Swersky, T. Pitassi, and C. Dwork. Learning fair representations. In *International Conference on Machine Learning*, pages 325–333, 2013.