A Statistical Perspective on Discovering Functional Dependencies in Noisy Data

Yunjia Zhang
yunjia@cs.wisc.edu
UW-Madison

Zhihan Guo
zhihan@cs.wisc.edu
UW-Madison

Theodoros Rekatsinas
thodrek@cs.wisc.edu
UW-Madison

ABSTRACT

We study the problem of discovering functional dependencies (FD) from a noisy data set. We adopt a statistical perspective and draw connections between FD discovery and structure learning in probabilistic graphical models. We show that discovering FDs from a noisy data set is equivalent to learning the structure of a model over binary random variables, where each random variable corresponds to a functional of the data set attributes. We build upon this observation to introduce FDX a conceptually simple framework in which learning functional dependencies corresponds to solving a sparse regression problem. We show that FDX can recover true functional dependencies across a diverse array of real-world and synthetic data sets, even in the presence of noisy or missing data. We find that FDX scales to large data instances with millions of tuples and hundreds of attributes while it yields an average F_1 improvement of 2× against state-of-the-art FD discovery methods.

KEYWORDS

Functional Dependencies; Structure Learning

ACM Reference Format:
Yunjia Zhang, Zhihan Guo, and Theodoros Rekatsinas. 2020. A Statistical Perspective on Discovering Functional Dependencies in Noisy Data. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (SIGMOD'20), June 14–19, 2020, Portland, OR, USA.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3318464.3389749

1 INTRODUCTION

Functional dependencies (FDs) are an integral part of data management. They are used in database normalization to reduce data redundancy and improve data integrity [15], and are critical for query optimization [20, 26, 28]. FDs are also helpful in data preparation tasks, such as data profiling and data cleaning [9, 40], and can also help guide feature engineering in machine learning pipelines [16]. Unfortunately, FDs are typically unknown and significant effort and domain expertise are required to identify them.

Many works have focused on automating FD discovery. Given a data instance, works from the database community [19, 25, 33] aim to enumerate all constraints that syntactically correspond to FDs and are not violated in the input data set (or are violated with some tolerance to accommodate for noisy data). On the other hand, data mining works [30, 31, 39] propose using information theoretic measures to identify FDs. Unfortunately, both approaches are limited either because they discover spurious constraints or because they do not scale to data sets with many attributes (see Section 5). The reason is that existing methods are by design prone to discover complex constraints, a behavior that can be formally explained if one views FD discovery via an information theoretic lens (see Section 2). *The main problem is that existing solutions are not designed to discover a parsimonious collection of FDs which is interpretable and can be readily used in downstream applications.* To address this problem, we adopt a statistical perspective of FD discovery and propose an FD discovery solution that is scalable and its output is interpretable without any tedious fine tuning.

Challenges. Inferring FDs from data observations poses many challenges. First, we need to identify an appropriate attribute order that captures the directionality of FDs in the data. This leads to a computational complexity that is exponential in the number of attributes in a data set. To address the exponential cost, existing methods rely on pruning to search over the lattice of attribute combinations. Pruning can either impose constraints on the number of attributes that participate in a constraint or can leverage information theoretic measures to filter constraints [25, 30]. Despite the use of pruning many existing methods are shown to exhibit poor scalability as the number of columns increases [25, 30].

Second, FDs capture deterministic relations between attributes. However, in real-world data sets missing or erroneous values introduce uncertainty to these relations. Noise

SIGMOD'20, June 14–19, 2020, Portland, OR, USA

ACM ISBN 978-1-4503-6735-6/20/06…$15.00
https://doi.org/10.1145/3318464.3389749

poses a challenge as it can lead to the discovery of spurious FDs or to low recall with respect to the true FDs in a data set. To deal with missing values and erroneous data, existing FD discovery methods focus on identifying *approximate* FDs, i.e., dependencies that hold for a portion a given data set. To identify approximate FDs, existing methods either limit their search over clean subsets of the data [34], which requires solving the expensive problem of error detection, or employ a combination of sampling methods assuming error models with strong biases such as random noise [25, 35]. These methods can be robust to noisy data. However, their performance, in terms of runtime and accuracy, is sensitive to factors such as sample sizes, prior assumptions on error rates, and the amount of records available in the input data set. This makes these methods hard to tune for data sets with varying number of attributes, records, and errors.

Finally, most dependency measures used in FD discovery, such as co-occurrence counts [25] or criteria based on mutual information [5] promote complex dependency structures [30]. The use of such measures leads to the discovery of spurious FDs in which the determinant set contains a large number of attributes. As we discuss later in our paper, this overfitting behavior stems directly by the use of measures such as entropy to measure dependencies across columns. Discovering large sets of FDs makes it hard for humans to interpret and validate the correctness of the discovered constraints. To avoid overfitting to spurious FDs existing methods rely on post-processing procedures to simplify the structure of discovered FDs or ranking based solutions. The most common approach is to identify *minimal FDs* [34]. An FD $X \rightarrow Y$ is said to be minimal if no subset of X determines Y. In many cases, this criterion is also integrated with search over the set of possible FDs for efficient pruning of the search space [25, 35]. Minimality can be effective, however, it does not guarantee that the set of discovered FDs will be parsimonious. As reported by Kruse et al., [25], hundreds of FDs for data sets with only tens of attributes.

Our Contributions. We propose FDX, a framework that relies on *structure learning* [24] to solve FD discovery. FDX leverages the strong dependencies that FDs introduce among attributes. We introduce a structured probabilistic model to capture these dependencies, and show that discovering FDs is equivalent to *learning the structure* of this model. A key contribution in our work is to model the distribution that FDs impose over pairs of records instead of the joint distribution over the attribute-values of the input data set. This approach is related to recent results on robust covariance estimation in the presence of corrupted data [6]. In summary, we are the first to use structure learning for principled, robust, easy-to-operationalize, and state-of-the-art FD-discovery, and our key technical contribution is to take the difference between tuple pairs for more robust and accurate structure learning.

FDX's model has *one binary random variable for each attribute* in the input data set and expresses correlations amongst random variables via a graph that relates random variables in a linear way. This linear model is inspired by standard results in the probabilistic modeling literature (see Section 4). We leverage linear dependencies to recover the FDs present in a data set. Given a noisy data set, FDX proceeds in two steps: First, it estimates the *undirected form* of the graph that corresponds to the FD model of the input data set. This is done by estimating the *inverse covariance matrix* of the joint distribution of the random variables that correspond to our FD model. Second, our FD discovery method finds a factorization of the inverse covariance matrix that imposes a sparse linear structure on the FD model, and thus, allows us to obtain parsimonious FDs.

We present an extensive experimental evaluation of FDX. First, we compare our method against state-of-the-art methods from both the database and data mining literature over a diverse array of real-world and synthetic data sets with varying number of attributes, domain sizes, records, and amount of errors. We find that FDX scales to large data instances with hundreds of attributes and yields an average F_1 improvement in discovering true FDs of more than $2\times$ compared to competing methods.

We also examine the effectiveness of FDX on downstream data preparation tasks. Specifically, we use FDX to profile real-world data sets and demonstrate how the dependencies that FDX discovers can (1) provide users with insights on the performance of automated data cleaning tools on the input data, and (2) can help users identify important features for predictive tasks associated with the input data. FDX is already deployed in several industrial use cases related to data profiling, including use cases in a major insurance company.

Outline. In Section 2, we discuss necessary background. In Section 3, we formalize the problem of FD discovery and provide an overview of FDX. In Section 4, we introduce the probabilistic model at the core of FDX and the structure learning method we use to infer its structure. In Section 5, we present the experimental evaluation of FDX. Finally, in Section 6 we discuss related work and conclude in Section 7.

2 BACKGROUND AND PRELIMINARIES

We review background material and introduce notation relevant to the problem we study in this paper. The topics discussed in this section aim to help the reader understand fundamental limitations of prior FD discovery works and basic concepts relevant to our proposed solution.

2.1 Functional Dependencies

We review the concept of FDs and related probabilistic interpretations adopted by prior works. We consider a data set D with a relational schema R. An FD $\mathbf{X} \rightarrow Y$ is a statement over the set of attributes $\mathbf{X} \subseteq R$ and an attribute $Y \in R$ denoting that an assignment to $\mathbf{X}$ uniquely determines the value of Y [15]. We consider $t_i[Y]$ to be the value of tuple $t_i \in D$ for attribute Y ; following a constraint-based interpretation, the FD $X \rightarrow Y$ holds iff for all pairs of tuples $t_i, t_j \in D$ we have that if $\bigwedge_{A \in \mathbf{X}} t_i[A] = t_j[A]$ then $t_i[Y] = t_j[Y]$. A functional dependency $\mathbf{X} \rightarrow Y$ is *minimal* if no subset of $\mathbf{X}$ determines Y in a given data set, and it is *non-trivial* if $Y \notin \mathbf{X}$.

Under the above constraint-based interpretation, to discover all FDs in a data set, it suffices to discover all minimal, non-trivial FDs. This interpretation assumes a closed-world and aims to find all syntactically valid FDs that hold in D. This constraint-based interpretation is adopted by several prior works [34], and as we discussed in Section 1 leads to the discovery of large numbers of FDs, i.e., overfitting. In addition, the interpretation of FDs as hard constraints leads to FD discovery solutions that are not robust to noisy data [25].

To address these limitations, a probabilistic interpretation of FDs can be adopted. Let each attribute $A \in R$ have a domain $V(A)$ and $V(\mathbf{X})$ be the domain of a set of attributes $\mathbf{X} = \{A_1, A_2, \ldots, A_k\} \subseteq R$ defined as $V(\mathbf{X}) = V(A_1) \times V(A_2) \times \cdots \times V(A_k)$. Also, assume that every instance D of R is associated with a probability density $f_R(D)$ such that these densities form a valid probability distribution P_R. Given the distribution P_R, we say that an FD $\mathbf{X} \rightarrow Y$, with $\mathbf{X} \subseteq R$ and $Y \in R$, holds if there is a function $\phi : V(\mathbf{X}) \rightarrow V(Y)$ with:

$$\forall \mathbf{x} \in V(\mathbf{X}) : P_R(Y = y | \mathbf{X} = \mathbf{x}) = \begin{cases} 1 - \epsilon, \text{ when } y = \phi(\mathbf{x}) \\ \epsilon, \text{ otherwise} \end{cases} \quad (1)$$

with ϵ being a small constant.

The above condition allows an FD to hold for *most tuples* allowing some violations. This equation captures the essence of approximate FDs used in multiple works [3, 19, 20, 25, 30]. Two core approaches are adopted in these works to discover approximate dependencies that satisfy:

(1) Use likelihood-based measures to find groups of attributes that satisfy Equation 1 [3, 19, 20, 25]. Typically these methods compute the approximate distribution (and likelihood) by considering co-occurrence counts between values of $(\mathbf{X}, Y)$ and normalizing those by counts of values of $\mathbf{X}$ [3, 20, 25]. For example, the likelihood of Equation 1 being satisfied can be estimated by aggregating the ratios $Count(x, y)/Count(x)$ for all values x in a finite instance (sample) D of R [3]. A likelihood of 1.0 means that the Equation 1 is satisfied.

(2) Rely on information theoretic measures [30] by considering the ratio $F(\mathbf{X}, Y) = \frac{H(Y) - H(Y|\mathbf{X})}{H(Y)}$ of the mutual information $H(Y) - H(Y|\mathbf{X})$ between Y and $\mathbf{X}$ (where $H(Y|\mathbf{X}) = \sum_{(\mathbf{x}, y)} P(\mathbf{X}, Y) \log P(Y|\mathbf{X})$ is the conditional entropy of Y given $\mathbf{X}$) and the entropy $H(Y)$ of Y. FDs satisfy that the ratio $F(\mathbf{X}, Y)$ is close to 1.0. Similar to the aforementioned approaches, these approaches require estimating the entropy $H(Y)$ and conditional entropy $H(Y|\mathbf{X})$ from a finite instance (sample) D of R by computing empirical co-occurrences across assignments of $\mathbf{X}$ and Y.

Both above approaches have a fundamental flaw: given a finite sample of tuples, as the number of attributes in $\mathbf{X}$ increases, it more likely that the empirical ratio $\hat{P}_R(\mathbf{x}|y) = |(\mathbf{x}, y)|/|\mathbf{x}|$ is 1.0, leading both aforementioned approaches to determine that Equation 1 is satisfied [1]. This behavior leads to *overfitting* to spurious dependencies and the discovery of complex (dense) structures across attributes. Intuitively, methods that rely on co-occurrence statistics or entropy-based measures capture marginal dependencies across attributes and not true conditional independencies as those implied by Equation 1 [24]. For the above reason, dependency discovery works that rely on the above techniques either employ filtering-based heuristics [3, 20, 25] or propose complex estimators [21, 30] to counteract overfitting.

2.2 Learning Parsimonious Structures

We also adopt a probabilistic interpretation of FDs but build upon structure learning methods in probabilistic graphical models [24] that directly discover conditional independencies to alleviate the overfitting problem. Graphical models are represented by a graph G where the nodes correspond to random variables and the absence of edges between nodes represent conditional independencies of variables. For example, a collection of independent variables corresponds to a collection of disconnected nodes, while a group of dependent variables may correspond to a clique [24].

To avoid overfitting, we need to learn graph structures that encode simple or low-dimensional distributions, i.e., the graph representing conditional independencies is sparse [14]. In fact, it was recently shown that one can provably recover the true dependency structure governing a data set by learning the sparsest possible conditional independencies that explain the data [38]. This property motivates our objective in this work of learning parsimonious models.

It is a standard result in statistical learning that one can learn the conditional independencies of a structured distribution by identifying the non-zero entries in the inverse *inverse covariance matrix* (a.k.a. precision matrix) $\Theta = \Sigma^{-1}$ of the data. The conditional dependencies amongst random variables are captured by the non-zero off-diagonal entries of Θ [24], and hence, zero off-diagonal entries in Θ represent conditional independencies amongst random variables.

[1] For information theoretic approaches, as $P(\mathbf{X}|Y)$ goes to 1.0, the condition entropy $H(Y|\mathbf{X})$ will be zero and $F(\mathbf{X}, Y)$ will be one.

One can learn the true conditional dependencies for a distribution by obtaining a sparse estimate of the inverse covariance matrix Θ from the observed data sample [47]. Many techniques have been proposed to obtain a sparse estimate for Θ [36] ranging from optimization methods [32] to efficient regression methods [14]. We are the first to show how these methods can be used to learn FDs. In addition, we propose an extension of these methods to enable robust FD discovery even in the presence of noisy data.

3 THE FDX FRAMEWORK

We formalize the problem of functional dependency discovery and provide an overview of FDX.

3.1 Problem Statement

We consider a relational schema R associated with a probability distribution P_R. We assume access to a noisy data set D' that follows schema R and is generated by the following process: first a clean data set D is sampled from P_R and a noisy channel model introduces noise in D to generate D'. We assume that D and D' have the same cells but cells in D' may have missing values or different values than their clean counterparts. We consider an *error* in D' to correspond to a cell c for which $D'(c) \neq D(c)$. We consider both incorrect and missing values. We assume that in expectation across all cells of the observed samples, a small fraction of cells in the data set, less than half, are corrupted. This assumption is necessary to recover the underlying structure of a distribution in the presence of corruptions [12]. This generative process is also considered in the database literature to model the creation of noisy data sets [42].

Given a noisy data instance D', our goal is to identify the functional dependencies that characterize the distribution P_R that from which the clean data set D was generated. Instead of modeling the structure of distribution P_R directly, we consider a different distribution with equivalent structure with respect to the FDs present in P_R: For any pair of tuples t_i and t_j sampled from P_R, we consider the random variable $I_{ij}[Y] = \mathbb{1}(t_i[Y] = t_j[Y])$ where $\mathbb{1}(\cdot)$ is the indicator function, and denote $t_i[\mathbf{X}]$ the value assignment for attributes $\mathbf{X}$ in tuple t_i. We say that $t_i[\mathbf{X}] = t_j[\mathbf{X}]$ iff $\bigwedge_{A\in\mathbf{X}} t_i[A] = t_j[A] =$ True. It is easy to see that an FD $\mathbf{X} \rightarrow Y$, with $\mathbf{X} \subseteq R$ and $Y \in R$, holds for P_R if for all pairs of tuples t_i, t_j in R we have the following condition for the distribution over random variables $I_{ij}[Y] = \mathbb{1}(t_i[Y] = t_j[Y])$:

$$\Pr(I_{ij}[Y] = 1 | t_i[\mathbf{X}] = t_j[\mathbf{X}]) = 1 - \epsilon \tag{2}$$

where ϵ is a small constant to ensure robustness against noise. This condition states that the random events $\bigwedge_{A\in\mathbf{X}}(t_i[A] = t_j[A])$ and $\mathbb{1}(t_i[Y] = t_j[Y])$ are deterministically correlated, which is equivalent to the FD $\mathbf{X} \rightarrow Y$. Under this interpretation, the problem of FD discovery corresponds to learning the structured dependencies amongst attributes of R that satisfy the above condition.

The reason we use this model is because estimating the inverse covariance (i.e., the dependencies) of the above distribution over the tuple differences and not the structure of P_R directly, yields FD discovery methods that are less sensitive to errors in the raw data (see Section 4.3). Beyond robustness to noise, this approach also enables us to identify dependencies over mixed distributions that may include categorical, numerical, or even textual data. The reason is that considering equality (or approximate equality) over attribute values enables us to represent any input as a binary data set with equivalent dependencies.

3.2 Solution Overview

An overview of our framework is shown in Figure 1. The input to our framework is a noisy data set and the output of our framework is a set of discovered FDs. The workflow of our framework follows three steps:

Data Set Transformation. First, we use the input data set D' and generate a collection of samples that correspond to outcomes of the random events $\bigwedge_{A\in\mathbf{X}}(t_i[A] = t_j[A]) =$ True and $t_i[Y] = t_j[Y]$. The output of this process is a new data set D_t that has one attribute for each attribute in D' but in contrast to D' it only contains binary values. We describe this step in Section 4.1.

Structure Learning. The transformed data output by the previous step corresponds to samples obtained by the model tuple pair-based model $\mathcal{M}$ described in Section 3.1. That is, data set D_t contains samples from the distribution of events $\bigwedge_{A\in\mathbf{X}}(t_i[A] = t_j[A]) =$ True and $t_i[Y] = t_j[Y]$. We learn the structure of $\mathcal{M}$ by obtaining a sparse estimate of its inverse covariance matrix from the samples in D_t. We describe our structure learning method in Section 4.2.

FD generation. Finally, we use a factorization of the estimated inverse covariance matrix to generate a collection of FDs. We describe this factorization in Section 4.2. The final output of our model is a collection of discovered FDs of the form $\mathbf{X} \rightarrow Y$ where $\mathbf{X} \subseteq R$ and $Y \in R$.

4 FD DISCOVERY IN FDX

We first introduce the probabilistic model that FDX uses to represent FDs and then describe our approach for learning its structure. Finally, we discuss how our approach compares to a naive application of structure learning to FD discovery.

4.1 The FDX Model

FDX's probabilistic model considers the FD interpretation described in Equation 2 and aims to capture the distribution of the random events $\bigwedge_{A\in\mathbf{X}}(t_i[A] = t_j[A])$ and $\mathbb{1}(t_i[Y] = t_j[Y])$.

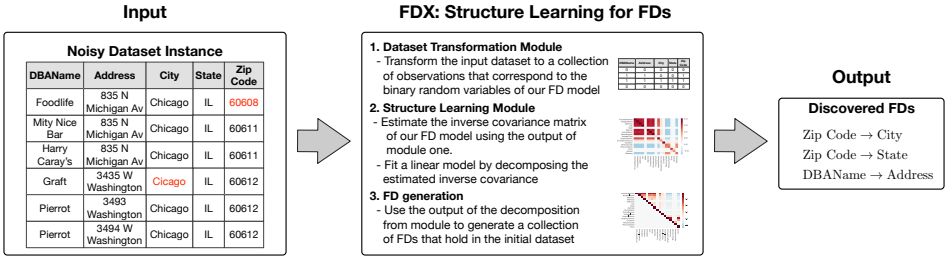


Figure 1: An overview of our structure learning framework for FD discovery

FDX's model consists of random variables that model these two random events. The edges in the model represent statistical dependencies that capture the relation in Equation 2.

We have one random variable per attribute in R. For each attribute $A \in R$, we denote $Z_A \in \{0, 1\}$ the random variable that captures the distribution that any two random tuples sampled from distribution P_R will have the same value for attribute A. In other words, to construct a sample for variable Z_A we first sample two random tuples (t_i, t_j) from P_R and then have that $Z_A = 1$ iff $t_i[A] = t_j[A]$. We also define $\mathbf{Z}$ to be the random vector containing variables Z_A for all attributes in R. An instance of $\mathcal{Z}$ corresponds to a binary vector capturing the equality across attribute values between two random tuples sampled from P_R. We now turn our attention to the dependencies over the binary random variables in $\mathbf{Z}$. For a set of attributes $\mathbf{X}$ let $\mathbf{Z}[\mathbf{X}]$ denote the corresponding values in vector $\mathbf{Z}$. Consider an FD $\mathbf{X} \rightarrow Y$. From Equation 2, we have that $\Pr(\mathbf{Z}[Y] = 1 | \mathbf{Z}[\mathbf{X}] = \mathbf{1}) = 1 - \epsilon$.

Our goal is to learn the structure of the model described above from samples corresponding to $\mathbf{Z}$. However, the dependencies across attributes in $\mathbf{Z}$ are V-structured (many-to-one), which makes structure learning an NP-hard problem [7]. We introduce two modeling assumptions to address this limitation and enable using structure learning methods with rigorous guarantees on correctness.

First, recent theoretical results in the statistical learning literature show that for *linear graphical models*, i.e., models that introduce linear dependencies between random variables, one can provably recover the correct structure from sample, even in the presence of corrupted samples [27]. In light of these results, we use a *linear structural equation model* to approximate the dependencies across attributes of the random vector $\mathbf{Z}$. We next describe the linear model we use and provide intuition why this approximation is reasonable.

To approximate the deterministic constraints introduced by FDs, we build upon techniques from *soft-logic* [2]. Soft logic allows continuous truth values in $[0, 1]$ instead of discrete truth values $0, 1$. Also, the Boolean logic operators are reformulated as: $A \wedge B = \max\{A + B - 1, 0\}$, $A \vee B = \min\{A + B, 1\}$, $A_1 \wedge A_2 \wedge \ldots A_k = \frac{1}{k} \sum_i A_i$, and $\neg A = 1 - A$.

Given the above, we denote $\hat{\mathbf{Z}}$ the $[0, 1]$-relaxed version of random vector $\mathbf{Z}$. We also consider that an FD $\mathbf{X} \rightarrow Y$ introduces the following linear dependency:

$$\hat{\mathbf{Z}}[Y] = \frac{1}{|\mathbf{X}|} \sum_{X_i \in \mathbf{X}} \hat{\mathbf{Z}}[X_i] \tag{3}$$

across coordinates of $\hat{\mathbf{Z}}$. This linear dependency approximates the condition in Equation 2 using soft-logic.

Second, to obtain a parsimonious model, we consider a global order of the random variables corresponding to the attributes in $\hat{\mathbf{Z}}$ and assume *acyclic dependencies*, i.e., our model assumes a global ordering over the schema attributes and only allows that for the relaxed condition in Equation 3 all attributes in $\mathbf{X}$ pre-ceed attribute Y in that ordering. This modeling choice is common when modeling dependencies over structured data [38, 45, 49]. Moreover, in our experimental evaluation in Section 5, we demonstrate that this assumption does not limit the effectiveness of FDX at discovering correct dependencies for real-world data (see Section 5.6.2).

Based on the aforementioned relaxed model, FDs force the relaxed random vector $\hat{\mathbf{Z}}$ to follow a *linear structured equation model*. It is easy to see that we can use a linear system of equations to express all linear dependencies of the form in Equation 3 that the attributes in $\hat{\mathbf{Z}}$ follow. We have:

$$\hat{\mathbf{Z}} = B^T \hat{\mathbf{Z}} + \epsilon, \tag{4}$$

where we assume that B is the *autoregression matrix* that captures the linear dependencies across attributes [27], $E[\epsilon] = 0$ and $\epsilon_j \perp\!\!\!\perp (\hat{Z}_{A_1}, \ldots, \hat{Z}_{A_{j-1}})$ for all j, where $\perp\!\!\!\perp$ denotes conditional independence. Since we assume that the coordinates in $\hat{\mathbf{Z}}$ follow a global order, matrix B is a strictly upper triangular matrix. This matrix is unknown and our goal is to infer its non-zero entries (i.e., structure) in order to recover the dependencies that are present in the input data set.

4.2 Structure Learning in FDX

Our structure learning algorithm follows from results in statistical learning theory. We build upon the recent results of Loh and Buehlmann [27] and Raskutti and Uhler [38] on learning the structure of linear structural models via

Algorithm 1: FD discovery with FDX

Input: A noisy relational dataset D' following schema R.
Output: A set of FDs of the form $\mathbf{X} \rightarrow Y$ on R.
Set $D_t \leftarrow$ `Transform`(D') (See Alg. 2);
Obtain an estimate $\hat{\Theta}$ of the inverse covariance matrix (e.g., using Graphical Lasso) where $\hat{\Theta} = UDU^T$ with U being upper triangular;
Set $\hat{B} = I - U$;
Set Discovered FDs $\leftarrow$ `GenerateFDs`(B) (See Alg. 3);
return Discovered FDs

inverse covariance estimation. Given a linear model as the one in Equation 4, it can be shown that the inverse covariance matrix $\Theta = \Sigma^{-1}$ of the model can be written as:

$$\Theta = \Sigma^{-1} = (I - B)\Omega^{-1}(I - B)^T \tag{5}$$

where I is the identity matrix, B is the autoregression matrix of the model, and $\Omega = \text{cov}[\epsilon]$ with $\text{cov}[\cdot]$ denoting the covariance matrix. This decomposition of Θ is commonly used in learning the structure of linear models [36, 38].

FD discovery in FDX proceeds as follows: First, we transform the sample data records in the input dataset D' to samples $\{\mathbf{Z}^i\}_{i=1}^N$ for the linear model in Equation 4 (see Algorithm 2); Second, we obtain an estimate $\hat{\Theta}$ of the inverse covariance matrix and factorize the estimate $\hat{\Theta}$ to obtain an estimate of the autoregression matrix $\hat{B}$ [38]; Third, we use the estimated matrix $\hat{B}$ to generate FDs (see Algorithm 3).

To find the structured dependencies we need to estimate Θ. We use the following approach: Suppose we have N observations and let S be the empirical covariance matrix of these observations. It is a standard result [32] that the sparse inverse covariance θ corresponds to a solution to the following optimization problem: $\min_{\Theta \succ 0} f(\Theta) := -\log\det(\Theta) + tr(S\Theta) + \lambda \|\Theta\|_1$ where we replace Θ with its factorization $\Theta = UDU^T$ with U being upper triangular. To find the solution of this problem for our setting, we use *graphical lasso* [14], as it is known to scale favorably to instances with a large number of variables, and hence, is appropriate for supporting data sets with a large number of attributes. Given the estimated inverse covariance matrix $\hat{\Theta}$ and its factorization we use the autoregression matrix $\hat{B}$ to generate FDs (see Algorithm 3).

We now turn our attention to how we transform the input dataset D' into a collection D_t of observations for the linear model of FDX (see Algorithm 2). We use the differences of pairs of tuples in dataset D' to generate D_t.

To construct the tuple pair samples in D_t, we use the following sampling procedure instead of drawing pairs of tuples uniformly at random: we iterate over all attributes in the dataset order the dataset with respect to the running attribute and perform a circular shift to construct pairs of tuples. We take the union of all tuple pairs constructed in this fashion. This heuristic allows us to increase obtain tuple pair samples that cover a wider range of attribute values, and hence, obtain a more representative sample D_t. The complexity of Algorithm 2 is quadratic in the number of attributes. Our method supports diverse data types (e.g., categorical data, real-values, text, binary data) as we can use different difference operations for each of these types.

The above structure learning procedure is guaranteed to recover the correct structure (i.e., identify correctly the non-zero entries) of matrix $\hat{B}$ with high probability as the number of samples in D_t goes to infinity and the number of errors in D is limited. These guarantees follow from [32] and [38].

Algorithm 2: Data Transformation

Input: A dataset D with n rows and k columns
Output: A dataset D_t with $n \cdot k$ rows and k columns
$A \leftarrow$ columns $[A_1, \ldots, A_k]$;
$D \leftarrow$ shuffle rows of D;
$D_t \leftarrow \emptyset$;
for $i = 1 : k$ **do**
  $D_i \leftarrow$ sort D by attribute A_i;
  $D_{i\_shift} \leftarrow$ circular shift of rows in D_i by 1;
  for $j = 1 : n$ **do**
    for $l = 1 : k$ **do**
      $D_t[(i-1)\cdot n + j, l] \leftarrow \mathbb{1}\left(D_i[j,l] = D_{i\_shift}[j,l]\right)$;
    end
  end
end
return D_t

Algorithm 3: FD generation

Input: An autoregression matrix B of dimensions $n \times m$, A schema R
Output: A collection of FDs
FDs $\leftarrow \emptyset$;
for $j = 1 : m$ **do**
  Set the column vector $b_j \leftarrow (B_{1,j}, B_{2,j}, \ldots, B_{j-1,j})$;
  $\mathbf{X} \leftarrow$ Take the attributes in R that corresponds to non-zero entries in b_j;
  Let A_j be the attribute in R with coordinate j ;
  if $\mathbf{X} \neq \emptyset$ **then**
    FDs $\leftarrow$ FDs $\cup \{\mathbf{X} \rightarrow A_j\}$;
  end
end
return FDs

4.3 Discussion

Recall that FDX performs structure learning over a sample constructed by taking the value differences over sampled

pairs of tuples from the raw data. There are two main important benefits that this approach offers in contrast to applying structure learning directly on the input data.

First, a standard maximum likelihood estimate of the covariance is very sensitive to the presence of outliers in the data set. The reason is that sample mean is used to estimate the covariance. However, the estimated mean can be biased due to errors in the data set. By sampling tuple differences, we effectively estimate the covariance of a transformed *zero-mean distribution* whose covariance has the same structure as the original distribution. *By fixing the mean to zero, covariance estimation is less sensitive to errors in the raw data.* This approach is rooted in robust statistics [6, 12]. We validate this experimentally in Section 5 where we show that FDX is more robust than standard Graphical Lasso.

Second, structure learning for FDX's model enjoys better sample complexity than structure learning on the raw data set. We focus on the case of discrete random variables to explain this argument. Let k be the size of the domain of the variables. The sample complexity of state-of-the-art structure learning algorithms is proportional to k^4 [47]. Our model restricts the domain of the random variables to be $k = 2$. At the same time, our transformation allows access to an increased amount of training data. Hence, our approach performs better than naive structure learning or other FD discovery methods when the sample size is small. We demonstrate this experimentally in Section 5.

5 EXPERIMENTS

We compare our approach against several FD discovery methods on different data sets. We seek to validate: (1) if structure learning enables accurate FD discovery (i.e,. high-prevision and high-recall), (2) what is the impact of different data characteristics on different FD discovery methods, (3) how robust FDX is to different tunable parameter settings, and (4) can we use the output of FDX to optimize downstream data preparation and data analytics pipelines. We also present synthetic micro-benchmarks to evaluate the robustness of FDX.

5.1 Experimental Setup

Methods. We consider four methods: (1) PYRO [25], a state-of-the-art FD discovery method in the database community that seeks to find all syntactically valid FDs in a data set. The code is released by the authors [2]. The scalability of the algorithm is controlled via an error rate hyper-parameter. (2) Reliable Fraction of Information (RFI) [30], the state-of-the-art FD discovery approach in Data Mining. RFI relies on an information theoretic score to find FDs and uses an approximation scheme to optimize performance. The approximation ratio is controlled by a hyper-parameter α. We evaluate RFI for $\alpha \in \{0.3, 0.5, 1\}$ where 1.0 corresponds to no approximation. The code is also released by the authors [3]. RFI discovers FDs for one attribute at a time and return a list of FDs in descending order with respect to RFI's score. For RFI, we keep the top-1 FD per attribute to obtain a parsimonious model and optimize its accuracy. To discover all FDs in a data set, we run the provided method once per attribute. (3) Graphical Lasso (GL), a structure learning algorithm for finding undirected structured dependences [47]. To find FDs, we perform a local graph search to find high-scored—we use the same score as RFI—directed structures. (4) TANE [19], another FD discovery algorithm that supports approximate FDs. The code is released by the authors [4]. To get approximate FDs on a noisy dataset, TANE uses a hyper-parameter that captures how much noise is expected; this parameter is left to its default setting if not specified in our experiments. (5) CORDS [20], a method to discover soft FDs and correlations. CORDS is using correlation-related statistics to identify FD dependencies between each pair of attributes. This baseline is a best-effort implementation of CORDS since the code is not available. All hyper-parameters are set according to [20].

Table 1: A summary of the benchmark data sets with known dependencies we use in our experiments.

Data set	Attributes	# FDs	# Edges in FDs
Alarm	37	24	45
Asia	8	6	8
Cancer	5	3	4
Child	20	15	20
Earthquake	5	3	8

Metrics. To account for partial discovery of FDs , we use Precision (P) defined as the fraction of correctly discovered edges that participate in true FDs by the total number of edges in discovered FDs; Recall (R) defined as the fraction of correctly discovered edges that participate in true FDs by the total number of true edges in the FDs of a data set; and F_1 is defined as $2PR/(P + R)$. For the synthetic data we consider five instances per setting. To ensure that we maintain the coupling amongst Precision, Recall, and F_1, we report the median performance. For all methods, we fine-tuned their hyper-parameters to optimize performance. In the case of PYRO we consulted the authors for this process. We also measure the end-to-end runtime for each method.

Evaluation Goals and Data Sets. First, we examine how accurately the different methods identify *true functional dependencies* in a data set. We consider functional dependencies that exist in the generating distribution of a data set

[2] https://github.com/HPI-Information-Systems/pyro/releases

[3] http://eda.mmci.uni-saarland.de/prj/dora/

[4] https://www.cs.helsinki.fi/research/fdk/datamining/tane/

Table 2: The different settings we consider for synthetic data sets. We use the description in parenthesis to denote each of these settings in our experiments.

Property	Settings
Noise Rate (n)	1% (Low), 30% (High)
Tuples (t)	1,000 (Small), 100,000 (Large)
Attributes (r)	8-16 (Small), 40-80 (Large)
Domain Cardinality (d)	64-216 (Small), 1,000-1,728 (Large)

Table 3: Real-world data sets for our experiments.

Data set	Tuples	Attributes
Australian	690	15
Hospital	1,000	17
Mammographic	830	6
NYPD	34,382	17
Thoraric	470	17
Tic-Tac-Toe	958	10

and use data sets with known functional dependencies. We use *benchmark data generation programs* that correspond to structured probabilistic models with functional dependencies (i.e., networks that exhibit deterministic dependencies). All data generators are obtained from a standard R package for Bayesian Networks[5] and are evaluated with their default settings. A summary of these data sets is shown in Table 1.

Second, we evaluate the above methods as we vary four key factors in the data: (1) Noise Rate (denoted by n). It stresses the robustness of FD discovery methods; (2) Number of Tuples (denoted by t). It affects the sample size available to the FD discovery methods; (3) Number of Attributes (denoted by r); It stresses the scalability of FD discovery methods; (4) Domain Cardinality (denoted by d) of the left-hand side $\mathbf{X}$ for an FD; It evaluates the sample complexity of FD methods. We consider 24 different setting combinations for these four dimensions (summarized in Table 2). For each setting we use a mixture of FDs $\mathbf{X} \rightarrow Y$ for which the cardinality of $\mathbf{X}$ ranges from one to three. We provide details on the synthetic data generation at the end of this section.

Finally, we evaluate the FD discovery methods on real-world data with naturally occurring errors that correspond to missing entries. For these data sets, we do not have access to the true FDs. We present a qualitative analysis of the discovered FDs as well as measurements on the runtime and the number of constraints discovered by each method. The data sets we use are benchmark data sets used to evaluate data cleaning and predictive analytics solutions[6]. Given this type of usage, we use these data sets to evaluate if FDX can help provide insights can (1) provide users with insights on the performance of automated data cleaning tools on the input data, and (2) can help users identify important features for predictive tasks associated with the input data. A summary of these data sets is provided in Table 3.

Synthetic Data Generation. We discuss our synthetic data generation process for completeness. The reader may safely continue with the next section. We follow the next process: Given a schema with r attributes our generator first assigns a global order to these attributes and splits the ordered attributes in consecutive attribute sets, whose size is between two and four (so that we obey the cardinality of the FD as we discussed above). Let $(\mathbf{X}, Y)$ be the attributes in such a split. Our generator samples a value v from the range associated with the setting for Domain Cardinality and assigns a domain to each attribute in $\mathbf{X}$ such that the cartesian product of the attribute values corresponds to that value. It also assigns the domain size of Y to be v.

We introduce FD dependencies as well as correlations in the splits obtained by the above process. For half of the $(\mathbf{X}, Y)$ groups generated via the above process, we introduce FD-based dependencies that satisfy the property in Equation 1. We do so by assigning each value $l \in \text{dom}(\mathbf{X})$ to a value $r_0 \in \text{dom}(Y)$ uniformly at random and generating t samples, where t is the value for the Tuples parameter. For the remainder of those groups we force the following conditional probability distribution: We assign each value $l \in \text{dom}(\mathbf{X})$ to a value $r_0 \in \text{dom}(Y)$. Then we generate t samples with $P(Y = r_0 \mid \mathbf{X} = l) = \rho$ and $P(Y \neq r_0 \mid \mathbf{X} = l) = \frac{1-\rho}{|\text{dom}(Y)-1|}$. Here, ρ is a hyper-parameter that is sampled uniformly at random from $[0, 0.85]$. This process allows us to mix FDs with other correlations, and hence, evaluate the ability of FD discovery mechanisms to differentiate between true FDs and strong correlations. Finally, to test how robust FD discovery algorithms are to noise, we randomly flip cells that correspond to attributes that participate in true FDs to a different value from their domain. The percentage of flipped cells is controlled by the Noise Rate setting.

5.2 Experiments on Known-Structure Data

We evaluate the performance of our approach and competing approaches on identifying FDs errors in all data sets with known structure. Table 4 summarizes the precision, recall, and F1-score obtained by different methods, and Table 5 summarizes their runtimes. For these data sets, we do not introduce noise given the inherent randomness of the data generation process.

[5] http://www.bnlearn.com/bnrepository/

[6] Many of the data sets are from the UCI repository; Hospital is from [40] and NYPD is the crime data set from the data portal of the city of New York.

Table 4: Evaluation on benchmark data sets with known functional dependencies.

Data set		FDX	GL	PYRO	TANE	CORDS	RFI(.3)	RFI(.5)	RFI(1.0)
Alarm	P	0.839	0.123	-	-	0.236	-	-	-
	R	0.578	0.867	-	-	0.778	-	-	-
	F_1	**0.684**	0.215	-	-	0.363	-	-	-
Asia	P	1.000	0.316	0.235	1.000	0.429	0.500	0.462	0.462
	R	0.500	0.750	0.500	0.125	0.750	0.750	0.750	0.750
	F_1	**0.667**	0.444	0.320	0.222	0.545	0.600	0.571	0.571
Cancer	P	1.000	0.375	1.000	0.000	0.000	0.571	0.571	0.571
	R	0.750	0.750	0.750	0.000	0.000	1.000	1.000	1.000
	F_1	**0.857**	0.500	**0.857**	0.000	0.000	0.727	0.727	0.727
Child	P	1.000	0.359	0.105	0.167	0.202	-	-	-
	R	0.450	0.700	1.000	0.400	0.900	-	-	-
	F_1	**0.667**	0.475	0.190	0.235	0.330	-	-	-
Earthquake	P	1.000	0.800	0.600	0.000	0.500	0.571	0.571	0.571
	R	1.000	1.000	0.750	0.000	0.750	1.000	1.000	1.000
	F_1	**1.000**	0.889	0.667	0.000	0.600	0.727	0.727	0.727

'-' method exceeds runtime limit (8 hours).

Table 5: Runtime (in seconds) of FD methods on benchmark data sets with known functional dependencies.

Data set	FDX	GL	PYRO	TANE	CORDS	RFI(.3)	RFI(.5)	RFI(1.0)
Alarm	2.468	2.827	-	-	**0.330**	-	-	-
Asia	0.388	0.213	1.598	0.090	**0.056**	13.009	15.231	15.336
Cancer	0.301	0.256	1.913	0.063	**0.047**	8.105	7.762	7.762
Child	1.128	0.468	217.748	**0.160**	0.169	-	-	-
Earthquake	0.366	0.181	3.337	**0.051**	0.065	7.038	7.767	6.601

'-' method exceeds runtime limit (8 hours).

As Table 4 shows, FDX consistently outperforms all other methods. In many cases, like Alarm, Asia, Child and Earthquake, we see improvements of 11 to 47 F1 points. We see that for data sets with few attributes and a small number of FDs (i.e., Asia, Cancer, and Earthquake) FDX achieves both high recall and high precision in all data sets despite the different distributional properties of each data set. For larger data sets (i.e., Alarm and Child), we see that FDX maintains its high precision but its recall drops. Nonetheless, FDX has a 47 points higher F_1-score than competing methods on the largest data set Alarm and is tied for the first place with PYRO on the second largest data set Cancer. In fact, TANE and RFI seem to be unable to obtain meaningful results for these cases. This performance is explained by the fact that FDX can be conservative in discovering FDs as it aims to learn a parsimonious dependency model. At the same time, Table 5 shows that FDX requires only a couple of seconds for the largest data set while it achieves relatively low run time for smaller data sets. The above results validate that FDX can identify true FDs effectively and efficiently.

We discuss the performance of individual competing methods. We start with PYRO. Recall that this method, finds all syntactically valid FDs in a data sample. Due to its design, we expect the recall of PYRO to be high but its precision limited. We see this behavior in the results shown in Table 4. We see that PYRO's recall is consistently higher than its precision, but in many cases the recall is not perfect. This is because PYRO is not as robust as other methods to noisy data.

We then focus on TANE. For most data sets, F_1-scores of TANE are consistently low in both recall and precision. We can see that for Cancer and Earthquake, no FDs are discovered by TANE. This is because TANE is finding equivalent row partitions which makes TANE not robust to noise. For CORDS, although run time is consistently low, we observe that it has unstable precision, recall and F_1-score. That is because CORDS only measures marginal dependencies and not conditional independence dependencies.

We turn our attention to RFI. RFI optimizes an information theoretic score to identify FDs. First, we find that RFI is significantly slower than all other methods (see Table 5) and it cannot terminate for data sets with many attributes. This performance is far from practical. For the data sets that it terminates we see that its F_1-score is better than PYRO but 10 to 28 points lower than FDX, with the precision of RFI being low. We attribute this performance to the RFI's score that tend to overfit the input sample and is not robust to noisy data. Finally, we do not observe quality differences as we vary the number of the approximation parameter. We also see that RFI is slower than FDX and its runtime increases dramatically for data sets with many attributes (to the extent that for Alarm it cannot terminate within eight hours).

Finally as for graphical lasso (GL), we see that it performs reasonably well in all data sets both with respect to F_1-score and runtime. However, we see that its precision is worse than FDX. This performance gap is due to the fact that unlike FDX, GL uses a non-robust covariance estimate.

Takeaway: The combination of structure learning methods with robust statistics is key to discovering true FDs in an effective and efficient manner.

5.3 Experiments with Synthetic Data

We perform a detailed evaluation of all FD discovery methods as we vary different key factors of the input data. To this end, we use the synthetic data described in Section 5.1. These data sets have varying characteristics summarized in Table 2.

Figure 2 shows the $F1$-score on four pairs of the synthetic data sets that we generate. Figure 2a, 2c, 2e and 2g show the results on high noise rate data sets, while Figure 2b, 2d, 2f and 2h show the results on low noise rate ones. We change the number of attributes r, number of tuples t and domain size d from large to small respectively. As shown, our FDX consistently outperforms all other baseline methods in terms of F_1-score in all settings. More importantly, we find out that our FDX is less affected by number of attributes and number of tuples compared with other baseline FD discovery methods. In detail, we find that FDX maintains good F_1-score for data sets with low amount of noises ($\leq 1\%$) with an

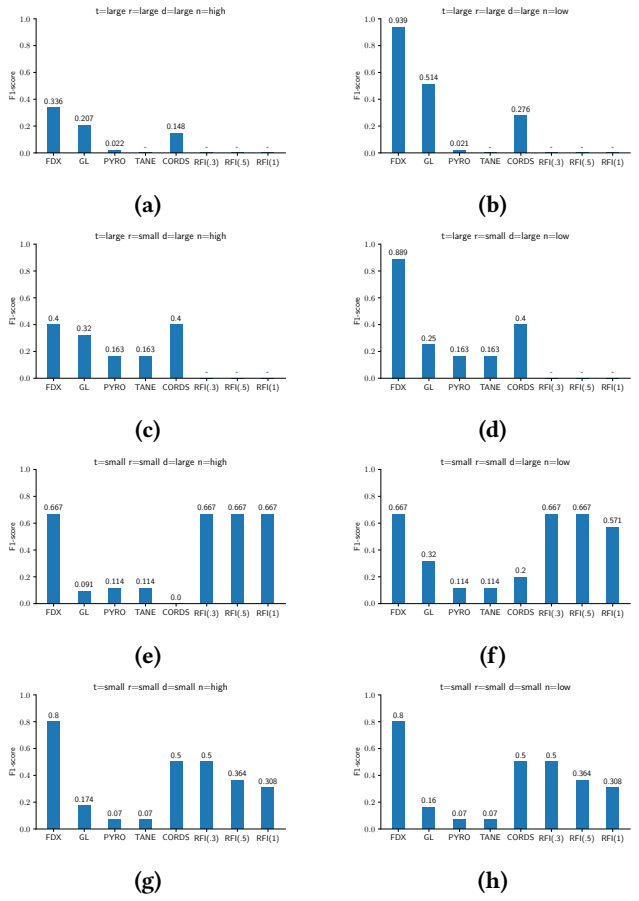


Figure 2: F_1-score of different methods on different synthetic settings

average of F_1-score of 0.823. For data sets with high noise rate, FDX still yields better results than competing methods.

To optimize performance of PYRO and TANE, we set their error rate hyper-parameter to the noise level for each data set. For the data set with large number of attributes r, TANE does not terminate. We observe that PYRO and TANE tend to generate near-complete FD graphs rather than sparse ones on synthetic data sets, which makes both PYRO and TANE have high recall, low precision, and low F_1-score. This behavior is compatible with their performance in our previous experiments with benchmark data.

As before we find that RFI exhibits poor scalability and in many cases it fails to terminate within 8 hours. When RFI terminates (shown in Figure 2e, 2f, 2g and 2h), we find that it exhibits good F_1-scores but still lower compared to FDX. We further investigated the performance of RFI for partial executions. Recall that due to the implementation of RFI, we have to run it for each attribute separately. We evaluated RFI's accuracy for each of the attributes processed within the 8-hour time window. Our findings are consistent with the aforementioned observation. The precision of RFI is high but its recall is lower than FDX.

Turning our attention to CORDS, we see again that its performance can vary significantly. For small instances, such as the instances in Figures 2 (g) and (h), we see that CORDS recovers the same dependencies as RFI (the entropy-based method) but for large instances, such as in Figures 2 (e) and (f), using the correlations to find FDs leads to overfitting and poor performance. This is because a small number of coordinates naturally limits the effect of overfitting to complex dependencies. This is why we see RFI's bias correcting estimator obtaining higher F_1-scores.

Finally, we see that the high sample complexity of structure learning on the raw input (see Section 4.3) leads to GL exhibiting low accuracy. This becomes more clear, if we compare the performance of GL with a large number of tuples to that with a small number of tuples while keeping other variables constant. We can see a consistent drop of performance when the data sample becomes limited.

Takeaway: Our evaluation on synthetic data verifies that the data transformation introduced in Section 4.1 enables FDX to be more robust to noisy data and allows for lower sample complexity. As a result, FDX can discovery FDs more accurately in the presence of noisy data. Furthermore, we find information theoretic measures exhibit higher sample complexity that pure statistical measures. This phenomenon is evident from the performance of RFI.

5.4 Experiments on Real-World Data

We evaluate different FD discovery methods against real-world data sets with naturally occurring errors that correspond to missing values.For our analysis, we use the data sets summarized in Table 3. As we discussed in Section 5.1, the true FDs are unknown for these data sets, and thus, we measure the runtime as well as the number of constraints discovered by each of the methods. Moreover, we manually inspect the constraints discovered by different methods and present a qualitative analysis.

Table 6 shows the runtime (in seconds) and the number of FDs discovered by each method. We first focus on runtime. As shown FDX, PYRO and TANE can scale to real-world noisy data instances with many attributes (e.g., NYPD). We see that for most data sets FDX terminates within a couple of seconds. The only exception is NYPD where FDX requires $\sim$ 400 seconds to terminate. This runtime is due to the data transformation introduced by Algorithm 2 that requires performing a self-join. Sampling methods can be used to further speed up this computation. We see that PYRO and TANE are also very efficient with most runtimes being below ten seconds. On the other hand, RFI has significant scalability

Table 6: Runtime (in seconds) and number of discovered FDs over real-world data sets with naturally occurring missing values.

Data set		FDX	GL	PYRO	TANE	CORDS	RFI(.3)	RFI(.5)	RFI(1.0)
Australian	time (sec)	0.38	0.46	10.44	0.12	0.07	621.59	985.93	2581.45
	# of FDs	4	14	1711	224	26	15	15	15
Hospital	time (sec)	1.75	0.59	2.65	0.16	0.13	6456.60	6603.16	6479.34
	# of FDs	10	16	434	655	39	16	16	16
Mammographic	time (sec)	0.24	0.18	1.47	0.07	0.04	4.73	5.52	5.02
	# of FDs	3	5	9	8	6	6	6	6
NYPD	time (sec)	447.48	1.43	5.49	3.96	0.84	-	-	-
	# of FDs	16	18	226	183	7	-	-	-
Thoracic	time (sec)	0.61	0.40	7.97	0.130	0.11	1938.56	3767.17	5528.76
	# of FDs	10	15	1066	53	13	17	17	17
Tic-Tac-Toe	time (sec)	1.02	0.28	9.04	0.10	0.09	39.99	59.57	70.48
	# of FDs	9	9	1168	98	18	10	10	10

'-' method exceeds runtime limit (8 hours).

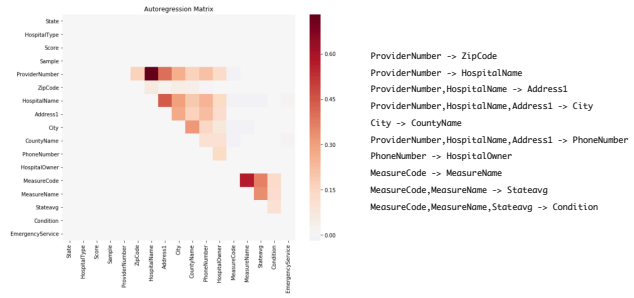


Figure 3: The autoregression matrix estimated by FDX for Hospital data set and the corresponding FDs.

issues when a data set has a large number of attributes. This performance makes RFI rather impractical for deployment in data pipelines.

We focus on the FDs discovered by the different methods. We see that FDX, GL, RFI, and CORDS always find a number of FDs that is at most equal to the number of attributes in the input data set. This behavior is expected as all these models are tailored towards finding a parsimonious set of FDs and for each attribute consider at most one FD that has this attribute as the determined attribute (i.e., on the right side). On the other hand, PYRO and TANE find hundreds of FDs for most data sets, as they find all syntactic FDs that hold in a given instance. Finally, we see that GL finds a similar number of constraints with FDX but there are cases where it discovers more constraints. This result is consistent with the behavior we observed in our previous experiments, i.e., that FDX is more conservative at reporting constraints. This behavior is desired in cases where a limited number of false positives is required. All these results are consistent with the FD interpretation adopted by each system. Based on these results, we argue that PYRO, TANE and RFI can be impractical in many cases.

```
HospitalName  -> ZipCode ( 0.6884822119510943)
HospitalName  -> HospitalOwner ( 0.7905101603249726)
HospitalName  -> Address1 ( 0.6841490007985284)
PhoneNumber  -> State ( 0.33850259042851694)
MeasureCode  -> Stateavg ( 0.7599899758330434)
HospitalName  -> PhoneNumber ( 0.68061335585621)
Condition, MeasureName  -> HospitalType ( 0.09808823042059128)
City  -> CountyName ( 0.7179703815912811)
MeasureName  -> MeasureCode ( 0.7884481625257015)
Sample  -> Score ( 0.2005127949958685)
MeasureCode  -> Condition ( 0.7896626996070244)
HospitalName  -> ProviderNumber ( 0.6864891049294678)
ProviderNumber  -> HospitalName ( 0.6896265931948304)
MeasureCode  -> MeasureName ( 0.7811219784869881)
HospitalName  -> City ( 0.6928075192148113)
ZipCode  -> EmergencyService ( 0.661887418552853)
```

Figure 4: The FDs discovered by RFI for Hospital.

We turn our attention to the quality of the FDs discovered by the competing approaches. We focus on the Hospital data set as it is easy to detect FDs via manual inspection. We consider the FDs discovered by FDX. A heatmap of the autoregression matrix of FDX's model and the corresponding FDs are shown in Figure 3. We find that the discovered FDs are meaningful. For example, we see that attributes 'Provider Number' and 'Hospital Name' determine most other attributes. We also see that 'Address' determines location-related attributes such as 'City'. We also find that attribute 'Measure Code' determines 'Measure Name' and that they both determine 'StateAvg'. In fact, there is an one-to-one mapping between 'MeasureCode' and 'MeasureName' while 'StateAvg' corresponds to the concatenation of the 'State' and 'Measure Code' attributes. The reader may wonder why the 'State' attribute is found to be independent of every other attribute. The reason is that hospital data set only contains two states with one appearing nearly 89% of time. Enforcing a sparse structure, FDX weakens the role of 'State' in deterministic relations. These results show that FDX can identify meaningful FDs in real-world data sets. We provide additional evidence in Section 5.5.

We now consider the constraints discovered by RFI. The results are consistent across all three alphas, so we pick the one with highest alpha (lower approximate rate). RFI outputs 16 FDs that are shown in Figure 4. The value in the parenthesis is the reliable fraction of information, the score proposed by RFI to select approximate FDs. After eliminating FDs with low score, we find that most of FDs discovered by RFI are also meaningful. However, it has the problem of overfitting to the data set. Specifically, for the FD 'ZipCode' $\rightarrow$ 'EmergencyService', this relation holds for the given data set instance, but does not convey any real-world meaning. We attribute this behavior to the fact that the domain of 'ZipCode' is really large while 'Emergency Service' only has a binary domain. This makes it more likely to observe a

Table 7: The F_1 score of AimNet and XGBoost for missing data imputation with random and systematic noise. We report the median accuracy for attributes that FDX identifies that participate in an FD (denoted by *w*) and attributes for which FDX identifies that do not participate in any FD (denoted by *w/o*).

Data set	Random Noise				Systematic Noise			
	AimNet		XGBoost		AimNet		XGBoost	
	w/o	w	w/o	w	w/o	w	w/o	w
Australian	0.41	**0.86**	0.34	**0.86**	0.42	**0.96**	0.34	**0.96**
Hospital	0.58	**1.0**	0.57	**0.97**	0.38	**1.0**	0.53	**0.99**
Mammogr.	0.63	**0.84**	0.54	**0.73**	0.44	**0.73**	0.42	**0.68**
NYPD	0.89	**0.93**	0.92	**0.94**	0.75	**0.76**	0.86	**0.90**
Thoracic	0.77	**0.82**	0.76	**0.83**	0.74	**0.91**	0.61	**0.91**
Tic-Tac-Toe	**0.6**	0.56	0.52	**0.55**	**0.48**	0.47	**0.57**	0.50

spurious FD when the number of data samples is limited. This finding matches RFI's performance for the synthetic data sets. For PYRO and TANE, we find that they discover hundreds of FDs, and hence, it is hard for a human to analyze. For instance, PYRO finds 24 FDs that determine 'Address1'.
Takeaway: We find that FDX can help users identify meaningful dependencies in real-world data with naturally occurring errors that correspond to missing values. Alternative approaches either do not scale to data sets with a large number of attributes or output an overwhelming number of constraints. The latter requires tedious inspection and fine-tuning by users to be valuable for downstream applications. In contrast to all prior approaches, structure learning offers a viable and practical solution to the problem of FD discovery.

5.5 Using FDX in Data Preparation

Summary. We examine if FDX's output can be useful for data profiling in data preparation pipelines. We consider two data preparation tasks: (1) automated data cleaning, and (2) feature engineering. For data cleaning, we demonstrate that *FDX can help predict if automated data cleaning will be effective*, and for feature engineering we demonstrate that *FDX can help identify important features for downstream predictive tasks without training any machine learning models.*

Results. We consider the data sets summarized in Table 3 and present the experiments we conduct for each of the aforementioned data preparation tasks as well as our findings. We consider the task of missing data imputation and two ML-based solutions to it: (1) AimNet, a new imputation method that relies on neural attention models to capture dependencies over the attributes of a data set [46], and (2) XGBoost (a method to shown to be very effective in [46]).

We build upon recent works that observe that in the presence of strong structured dependencies automated data cleaning can be effective [17, 40] and perform the following experiment: For each data set in Table 3, we separate its attributes into two groups (1) attributes that participate in an FD based on FDX's output, and (2) attributes that are independent according to FDX. We measure the median imputation accuracy for each group for AimNet and XGBoost and *examine if the constraints discovered by FDX can be used as a proxy to identify if automated cleaning will be accurate.*

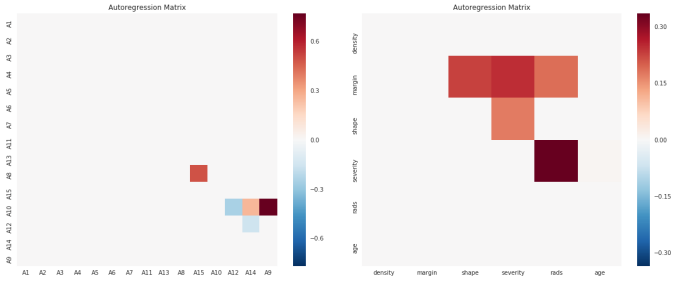
The results are summarized in Table 7. We see that in most cases, the accuracy of data imputation is higher when the target attribute participates in a dependency identified by FDX. This pattern holds for both AimNet and XGBoost, which provides evidence that FDX can be used as an effective data profiling mechanism regardless of the model used for data cleaning. In fact, FDX is already being used in industrial use cases as a profiling tool in data preparation pipelines.

For feature engineering, we focus on the Australian Credit Approval and the Mammography data sets. For Australian, the attributes are anonymized and the target attribute is $A15$. For Mammography, the target attribute is 'severity'. Figure 5 shows the autoregressive matrices recovered by FDX. As depicted, for Australian FDX finds that attribute $A8$ determines the target attribute $A15$. After investigating the literature we find reports [37] which state that indeed $A8$ is the most informative feature for the corresponding prediction task. In fact, this report evaluates several feature-ranking methods that all rank $A8$ as the most important feature for this task. For Mammography, FDX finds that the mass 'margin' and 'shape' determine the 'severity' of a mass (i.e., the target attribute) and that 'severity' determines the BI-RADS assessment (attribute 'rads'). We find publications in the medical domain [48] as well as a textbook in cancer medicine [18] which state that "the most significant features that indicate whether a mass is benign or malignant are its shape and margins", a fact that is indeed recovered by FDX. Moreover, the publication associated with this data set [29] explains that the BI-RADS assessment records the assessment of medical doctors and is predictive of malignancy. Notice that FDX finds the correct directionality between the severity of a mass and the BI-RADS assessment.

5.6 Hyper-parameter Analysis

We examine FDX's robustness against different hyperparameter settings. We report results for: (1) different sparsity settings, and (2) different column ordering methods.

5.6.1 Sparsity Setting. We present the results of FDX on known-structure benchmark data set with different sparsity settings. As we see in Table 8, there is a constant drop on



(A) Australian Credit Approval; A15 is the goal attribute

(B) Mammography; Severity is the goal attribute

Figure 5: The autoregression matrix estimated by FDX for Australian Credit Approval and Mammography.

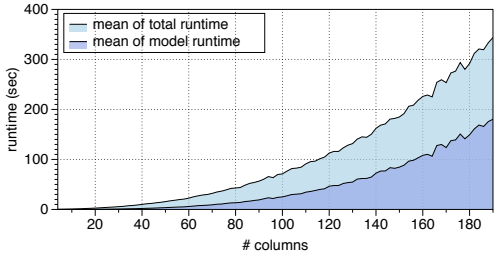


Figure 6: Columns-wise scalability of FDX.

Table 8: Evaluation on benchmark data sets with different sparsity setting for FDX.

Data set		0	.002	.004	.006	.008	.010
Alarm	Precision	0.839	0.723	0.694	0.640	0.627	0.632
	Recall	0.578	0.756	0.755	0.711	0.711	0.689
	F_1-score	0.684	0.739	0.723	0.673	0.667	0.659
	# of FDs	31	30	30	30	30	28
Asia	Precision	1.000	0.714	0.667	0.667	0.667	0.800
	Recall	0.500	0.625	0.500	0.500	0.500	0.500
	F_1-score	0.667	0.444	0.571	0.571	0.571	0.615
	# of FDs	7	6	5	5	5	4
Cancer	Precision	1.000	1.000	0.000	0.000	0.000	0.000
	Recall	0.750	0.250	0.000	0.000	0.000	0.000
	F_1-score	0.857	0.400	0.000	0.000	0.000	0.000
	# of FDs	3	1	0	0	0	0
Child	Precision	1.000	0.778	0.696	0.727	0.696	0.714
	Recall	0.450	0.700	0.800	0.800	0.800	0.75
	F_1-score	0.667	0.737	0.744	0.762	0.744	0.732
	# of FDs	14	18	17	17	17	15
Earthquake	Precision	1.000	1.000	1.000	1.000	1.000	1.000
	Recall	1.000	0.750	0.750	0.750	0.750	0.750
	F_1-score	1.000	0.857	0.857	0.857	0.857	0.857
	# of FDs	4	3	3	3	3	3

'-' method exceeds runtime limit (8 hours).

Table 9: Evaluation on FDX using known-structure data sets with different column ordering methods

Data set		heuristic	natural	amd	colamd	metis	nesdis
Alarm	P	0.839	0.839	0.839	0.839	0.867	0.839
	R	0.578	0.578	0.578	0.578	0.578	0.578
	F_1	0.684	0.684	0.684	0.684	**0.693**	0.684
Asia	P	1.000	1.000	0.800	0.800	0.800	0.800
	R	0.500	0.500	0.500	0.500	0.500	0.500
	F_1	**0.667**	**0.667**	0.615	0.615	0.615	0.615
Cancer	P	1.000	1.000	0.500	1.000	1.000	1.000
	R	0.750	0.750	1.000	0.750	0.750	0.750
	F_1	**0.857**	**0.857**	0.667	**0.857**	**0.857**	**0.857**
Child	P	1.000	1.000	1.000	1.000	1.000	1.000
	R	0.450	0.450	0.450	0.450	0.450	0.450
	F_1	**0.667**	**0.667**	**0.667**	**0.667**	**0.667**	**0.667**
Earthquake	P	1.000	1.000	0.800	0.444	0.800	0.800
	R	1.000	1.000	1.000	1.000	1.000	1.000
	F_1	**1.000**	**1.000**	0.889	0.615	0.889	0.889

number of FDs along as we increase sparsity. For large data sets (i.e. Child and Alarm), as we increase the sparsity, we observe an increasing trend in F_1-score followed by a decreasing trend. This finding is in line with our claim that for a large data set with many attributes, we should apply some sparsity to achieve the parsimonious graph structure.

5.6.2 Column Ordering. We report results of FDX on known-structure data sets under different column ordering methods in Table 9. The decomposition we consider corresponds to a version of the Cholesky decomposition. There are many common heuristics to determine variable orderings for that decomposition, and hence, different variable ordering for the linear structural model we consider. In all previous experiments we use the use the minimum-degree ordering heuristic [11] to obtain a sparsity-inducing decomposition. To evaluate performance of FDX on different ordering, we consider different ordering heuristics used in standard Cholesky decomposition packages [11]. As we can see in Table 9, FDX is not sensitive to ordering method: FDX with heuristic and natural ordering (i.e., using the default ordering of the data set) generates the best results for most data sets.

5.7 Micro-benchmark Results

We report micro-benchmarking results: (1) we evaluate the scalability of FDX and demonstrate its quadratic computational complexity with respect to number of attributes; (2) evaluate the effect of increasing noise rates on the performance of FDX.

5.7.1 Column-wise Scalability. Based on our discussion in Section 4, FDX exhibits quadratic complexity instead of exponential complexity with respect to the number of columns in a data set. We experimentally demonstrate FDX's scalability. We generate a collection of synthetic data sets where we keep all settings fixed except for the number of attributes, which we range from 4 to 190 with a increase step of two. For each number of columns, we generate five data sets and calculate the average runtime for each columns size. In addition, we log both the total runtime (including data loading and data transformation) and the structure learning runtime. The results are shown in Figure 6 and validate the quadratic scalability of FDX as the number of attributes increase.

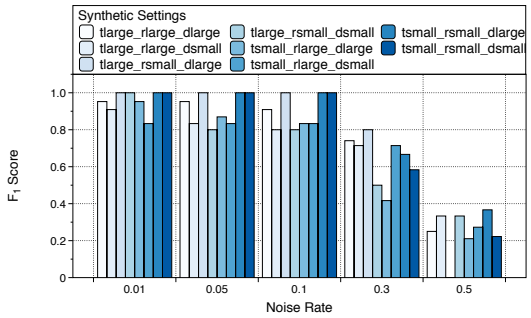


Figure 7: Effect of noise on FDX's performance. The data set names indicate the setting used (see Table 2).

5.7.2 Effect of Increasing Noise Rates. We evaluate how FDX performs as the noise rate increases. For this experiment we generate a new set of synthetic data sets and measure the performance of FDX for noise rates in $\{0.01, 0.1, 0.3, 0.5\}$. We report the median F_1 score in Figure 7. As expected, the performance of FDX deteriorates as the noise increases, however, FDX is shown to be robust to high error rates.

6 RELATED WORK

FD discovery is a critical problem in many data management applications. These applications include data cleaning [4, 9, 10, 40, 41, 44], schema normalization [15], and query optimization [20]. Numerous algorithms have been proposed for discovering syntactically valid FDs in a data set [1, 19, 25, 30, 34, 35]. We review the works related to ours:

Noisy FD Discovery. For noisy FD discovery, proposed solutions aim to identify FDs that hold approximately [19, 25, 30]. The works of Kruse and Naumann [25] and the work of Huhtala et al. [19] set the maximum rows that approximate FDs can violate. These works rely on co-occurrences to identify FDs and are agnostic to the type of errors in the data. Due to their design they require significant tuning that can be counter intuitive to the user. The work of Mandros et al. [30] is also agnostic to the types of errors in a data set and relies on bounding an entropy-based score to obtain approximate solutions. Finally, CORDS consider correlations to obtain soft FDs [20]. However, the co-occurrence measures considered in CORDS discover marginal dependencies and not conditional independencies that correspond to true FDs. There are also works that focus explicitly on missing values. Specifically, the work by Berti-Equille et al. [3] leverages likelihood-based measures (computed by considering value co-occurrences) to identify true FDs in the data.

All these methods that rely either on co-occurrences or likelihood- or entropy-based methods can overfit to spurious, complex functional dependencies. To counteract this flaw all aforementioned methods rely on complex filtering or estimation procedures that can be hard for users to optimize. On the other hand, the rigorous statistical grounding of FDX provides a FD discovery solution that requires minimal tuning as shown in Section 5.

Discovery of Other Constraints. There are also many works that consider discovering other types of constraints and are not limited to FDs alone [8, 13, 21–23].

For instance, there is work that considers discovering key constraints under inconsistent data [22, 23]. These works rely on axiom systems for constraints and propose algorithms to discover approximate certain keys, i.e., keys with potentially erroneous values that can identify tuples but may also have violating values. Other works [8, 21] focus on more general forms of constraints. Chu et al. [8] focus on the discovery of denial constraints and present a predicate-based algorithm that calculates evidence sets of constraint satisfaction over the input data. More recently, Kenig et al. [21] focused on the problem of discovering multi-valued dependencies over noisy data. The methods in [21] rely on entropy-based measures to score candidate constraints and are related to heuristic, search-based structure learning methods used in directed graphical models [24]. Such entropy-based approaches to structure learning exhibit an inherent tendency to overfit spurious relationships (see Section 2). For this reason works such as [43] rely on complex and expensive search procedures to find valid constraints. Our work demonstrates that structure learning methods based on the inverse covariance are simpler and come with rigorous statistical guarantees.

7 CONCLUSIONS

We introduced FDX, a structure learning framework to solve the problem of FD discovery in relational data. A key result in our work is to model the distribution that FDs impose over pairs of records instead of the joint distribution over the attribute-values of the input dataset. Specifically, we introduce a method that convert FD discovery to a structure learning problem over a linear structured equation model. We empirically show that FDX outperforms state-of-the-art FD discovery methods and can produce meaningful FDs that are useful for downstream data preparation tasks.

8 ACKNOWLEDGEMENTS

This work was supported by Amazon under an ARA Award, by NSF under grant IIS-1755676, and by DARPA under grant ASKE HR00111990013. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views, policies, or endorsements, either expressed or implied, of DARPA or the U.S. Government.

REFERENCES

[1] Jalal Atoum, Dojanah Bader, and Arafat Awajan. 2008. Mining functional dependency from relational databases using equivalent classes and minimal cover. In *Journal of Computer Science*. Citeseer.

[2] Stephen H. Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. 2017. Hinge-loss Markov Random Fields and Probabilistic Soft Logic. *J. Mach. Learn. Res.* 18, 1 (Jan. 2017), 3846–3912.

[3] Laure Berti-Équille, Hazar Harmouch, Felix Naumann, Noël Novelli, and Saravanan Thirumuruganathan. 2018. Discovery of Genuine Functional Dependencies from Relational Data with Missing Values. *Proc. VLDB Endow.* 11, 8 (April 2018), 880–892. https://doi.org/10.14778/3204028.3204032

[4] Philip Bohannon, Wenfei Fan, Floris Geerts, Xibei Jia, and Anastasios Kementsietsidis. 2007. Conditional functional dependencies for data cleaning. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*. IEEE, 746–755.

[5] Roger Cavallo and Michael Pittarelli. 1987. The Theory of Probabilistic Databases. In *Proceedings of the 13th International Conference on Very Large Data Bases (VLDB '87)*. 71–81.

[6] Yu Cheng, Ilias Diakonikolas, Rong Ge, and David P. Woodruff. 2019. Faster Algorithms for High-Dimensional Robust Covariance Estimation. In *Proceedings of the Thirty-Second Conference on Learning Theory (Proceedings of Machine Learning Research)*, Alina Beygelzimer and Daniel Hsu (Eds.), Vol. 99. Phoenix, USA, 727–757.

[7] David Maxwell Chickering, David Heckerman, and Christopher Meek. 2004. Large-Sample Learning of Bayesian Networks is NP-Hard. *J. Mach. Learn. Res.* 5 (Dec. 2004), 1287–1330.

[8] Xu Chu, Ihab F Ilyas, and Paolo Papotti. 2013. Discovering denial constraints. *Proceedings of the VLDB Endowment* 6, 13 (2013), 1498–1509.

[9] X. Chu, I. F. Ilyas, and P. Papotti. 2013. Holistic data cleaning: Putting violations into context. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*. 458–469.

[10] Michele Dallachiesa, Amr Ebaid, Ahmed Eldawy, Ahmed Elmagarmid, Ihab F Ilyas, Mourad Ouzzani, and Nan Tang. 2013. NADEEF: a commodity data cleaning system. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. ACM, 541–552.

[11] Timothy A Davis. User Guide for CHOLMOD: a sparse Cholesky factorization and modification package. (????).

[12] Ilias Diakonikolas, Gautam Kamath, Daniel M. Kane, Jerry Li, Ankur Moitra, and Alistair Stewart. 2017. Being Robust (in High Dimensions) Can Be Practical. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70 (ICML'17)*. JMLR.org, 999–1008. http://dl.acm.org/citation.cfm?id=3305381.3305485

[13] Wenfei Fan, Floris Geerts, Jianzhong Li, and Ming Xiong. 2010. Discovering conditional functional dependencies. *IEEE Transactions on Knowledge and Data Engineering* 23, 5 (2010), 683–698.

[14] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. 2008. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics* 9, 3 (2008), 432–441.

[15] Hector Garcia-Molina, Jennifer Widom, and Jeffrey D. Ullman. 1999. *Database System Implementation*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

[16] Luca M. Ghiringhelli, Jan Vybiral, Sergey V. Levchenko, Claudia Draxl, and Matthias Scheffler. 2015. Big Data of Materials Science: Critical Role of the Descriptor. *Phys. Rev. Lett.* 114 (2015), 105503. Issue 10.

[17] Alireza Heidari, Joshua McGrath, Ihab F. Ilyas, and Theodoros Rekatsinas. 2019. HoloDetect: Few-Shot Learning for Error Detection. In *Proceedings of the 2019 International Conference on Management of Data (SIGMOD '19)*. ACM, New York, NY, USA, 829–846. https://doi.org/10.1145/3299869.3319888

[18] Waun Ki Hong, William Hait, James F Holland, Donald W Kufe, and Raphael E Pollock. 2010. *Holland-Frei Cancer Medicine*. PMPH-USA.

[19] Ykä Huhtala, Juha Kärkkäinen, Pasi Porkka, and Hannu Toivonen. 1999. TANE: An Efficient Algorithm for Discovering Functional and Approximate Dependencies. *Comput. J.* 42, 2 (1999), 100–111.

[20] Ihab F. Ilyas, Volker Markl, Peter Haas, Paul Brown, and Ashraf Aboulnaga. 2004. CORDS: Automatic Discovery of Correlations and Soft Functional Dependencies. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data (SIGMOD '04)*. ACM, New York, NY, USA, 647–658. https://doi.org/10.1145/1007568.1007641

[21] Batya Kenig, Pranay Mundra, Guna Prasad, Babak Salimi, and Dan Suciu. 2019. Mining Approximate Acyclic Schemes from Relations. *CoRR* abs/1911.12933 (2019). arXiv:1911.12933 http://arxiv.org/abs/1911.12933

[22] Henning Köhler, Sebastian Link, and Xiaofang Zhou. 2015. Possible and Certain SQL Keys. *Proc. VLDB Endow.* 8, 11 (July 2015), 1118âĂŞ1129.

[23] Henning Kohler, Sebastian Link, and Xiaofang Zhou. 2016. Discovering Meaningful Certain Keys from Incomplete and Inconsistent Relations. (2016).

[24] Daphne Koller and Nir Friedman. 2009. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press.

[25] Sebastian Kruse and Felix Naumann. 2018. Efficient discovery of approximate dependencies. *Proceedings of the VLDB Endowment* 11, 7 (2018), 759–772.

[26] Viktor Leis, Andrey Gubichev, Atanas Mirchev, Peter Boncz, Alfons Kemper, and Thomas Neumann. 2015. How Good Are Query Optimizers, Really? *Proc. VLDB Endow.* 9, 3 (Nov. 2015), 204–215. https://doi.org/10.14778/2850583.2850594

[27] Po-Ling Loh and Peter Bühlmann. 2014. High-dimensional learning of linear causal networks via inverse covariance estimation. *The Journal of Machine Learning Research* 15, 1 (2014), 3065–3105.

[28] Guy Lohman. Is query optimization a âĂIJsolvedâĂİ problem.

[29] Simone A. Ludwig. 2010. Prediction of Breast Cancer Biopsy Outcomes Using a Distributed Genetic Programming Approach. In *Proceedings of the 1st ACM International Health Informatics Symposium (IHI '10)*. ACM, New York, NY, USA, 694–699. https://doi.org/10.1145/1882992.1883099

[30] Panagiotis Mandros, Mario Boley, and Jilles Vreeken. 2017. Discovering reliable approximate functional dependencies. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM.

[31] Panagiotis Mandros, Mario Boley, and Jilles Vreeken. 2018. Discovering Reliable Dependencies from Data: Hardness and Improved Algorithms. In *ICDM*. IEEE Computer Society, 317–326.

[32] Nicolai Meinshausen, Peter Bühlmann, et al. 2006. High-dimensional graphs and variable selection with the lasso. *The annals of statistics* 34, 3 (2006), 1436–1462.

[33] Thorsten Papenbrock, Jens Ehrlich, Jannik Marten, Tommy Neubert, Jan-Peer Rudolph, Martin Schönberg, Jakob Zwiener, and Felix Naumann. 2015. Functional Dependency Discovery: An Experimental Evaluation of Seven Algorithms. *Proc. VLDB Endow.* 8, 10 (June 2015), 1082–1093.

[34] Thorsten Papenbrock, Jens Ehrlich, Jannik Marten, Tommy Neubert, Jan-Peer Rudolph, Martin Schönberg, Jakob Zwiener, and Felix Naumann. 2015. Functional dependency discovery: An experimental evaluation of seven algorithms. *Proceedings of the VLDB Endowment* 8, 10 (2015), 1082–1093.

[35] Thorsten Papenbrock and Felix Naumann. 2016. A hybrid approach to functional dependency discovery. In *Proceedings of the 2016 International Conference on Management of Data*. ACM, 821–833.

[36] Mohsen Pourahmadi. 2011. Covariance Estimation: The GLM and Regularization Perspectives. *Statist. Sci.* 26, 3 (08 2011), 369–387.

[37] Pannir Rajaleximi, Mohammed Ahmed, and Ahmed Alenezi. 2019. Feature Selection using Optimized Multiple Rank Score Model for Credit Scoring. *International Journal of Intelligent Engineering and Systems* 12 (04 2019), 74–84.

[38] Garvesh Raskutti and Caroline Uhler. 2018. Learning directed acyclic graph models based on sparsest permutations. *Stat* 7, 1 (2018), e183.

[39] Matthew Reimherr and Dan L. Nicolae. 2013. On Quantifying Dependence: A Framework for Developing Interpretable Measures. *Statist. Sci.* 28 (02 2013).

[40] Theodoros Rekatsinas, Xu Chu, Ihab F. Ilyas, and Christopher Ré. 2017. HoloClean: Holistic Data Repairs with Probabilistic Inference. *Proc. VLDB Endow.* 10, 11 (2017).

[41] El Kindi Rezig, Mourad Ouzzani, Ahmed K. Elmagarmid, Walid G. Aref, and Michael Stonebraker. 2019. Towards an End-to-End Human-Centric Data Cleaning Framework. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics (HILDA'19)*. ACM, New York, NY, USA, Article 1, 7 pages. https://doi.org/10.1145/3328519.3329133

[42] Christopher De Sa, Ihab F. Ilyas, Benny Kimelfeld, Christopher Ré, and Theodoros Rekatsinas. 2019. A Formal Framework for Probabilistic Unclean Databases. In *International Conference on Database Theory, ICDT 2019*.

[43] Philipp Schirmer, Thorsten Papenbrock, Sebastian Kruse, Felix Naumann, Dennis Hempfing, Torben Mayer, and Daniel Neuschäfer-Rube. 2019. DynFD: Functional Dependency Discovery in Dynamic Datasets. In *EDBT*. OpenProceedings.org, 253–264.

[44] Saravanan Thirumuruganathan, Laure Berti-Equille, Mourad Ouzzani, Jorge-Arnulfo Quiane-Ruiz, and Nan Tang. 2017. Uguide: User-guided discovery of fd-detectable errors. In *Proceedings of the 2017 ACM International Conference on Management of Data*. ACM, 1385–1397.

[45] Kostas Tzoumas, Amol Deshpande, and Christian S. Jensen. 2011. Lightweight Graphical Models for Selectivity Estimation Without Independence Assumptions. *PVLDB* 4, 11 (2011), 852–863.

[46] Richard Wu, Aoqian Zhang, Ihab Ilyas, and Theodoros Rekatsinas. 2020. Attention-based Learning for Missing Data Imputation in HoloClean. In *Proceedings of Machine Learning and Systems 2020*. 307–325.

[47] Shanshan Wu, Sujay Sanghavi, and Alexandros G Dimakis. 2018. Sparse Logistic Regression Learns All Discrete Pairwise Graphical Models. *arXiv preprint arXiv:1810.11905* (2018).

[48] Yirong Wu, Oguzhan Alagoz, Mehmet U. S. Ayvaci, Alejandro Munoz del Rio, David J. Vanness, Ryan Woods, and Elizabeth S. Burnside. 2013. A Comprehensive Methodology for Determining the Most Informative Mammographic Features. *Journal of Digital Imaging* 26, 5 (01 Oct 2013), 941–947.

[49] Zongheng Yang, Eric Liang, Amog Kamsetty, Chenggang Wu, Yan Duan, Xi Chen, Pieter Abbeel, Joseph M. Hellerstein, Sanjay Krishnan, and Ion Stoica. 2019. Selectivity Estimation with Deep Likelihood Models. *CoRR* abs/1905.04278 (2019).