SCODED: Statistical Constraint Oriented Data Error Detection

Jing Nathan Yan*
Cornell University
jy858@cornell.edu

Oliver Schulte
Simon Fraser University
oschulte@sfu.ca

MoHan Zhang
Simon Fraser University
mohan_zhang@sfu.ca

Jiannan Wang
Simon Fraser University
jnwang@sfu.ca

Reynold Cheng
The University of Hong Kong
ckcheng@cs.hku.hk

ABSTRACT

Statistical Constraints (SCs) play an important role in statistical modeling and analysis. This paper brings the concept to data cleaning and studies how to leverage SCs for error detection. SCs provide a novel approach that has various application scenarios and works harmoniously with downstream statistical modeling. Entailment relationships between SCs and integrity constraints provide analytical insight into SCs. We develop SCODED, an SC-Oriented Data Error Detection system, comprising two key components: (1) *SC Violation Detection*: checks whether an SC is violated on a given dataset, and (2) *Error Drill Down*: identifies the top-k records that contribute most to the violation of an SC. Experiments on synthetic and real-world data show that SCs are effective in detecting data errors that violate them, compared to state-of-the-art approaches.

ACM Reference Format:
Jing Nathan Yan, Oliver Schulte, MoHan Zhang, Jiannan Wang, and Reynold Cheng. 2020. SCODED: Statistical Constraint Oriented Data Error Detection. 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/10.1145/3318464.3380568

1 INTRODUCTION

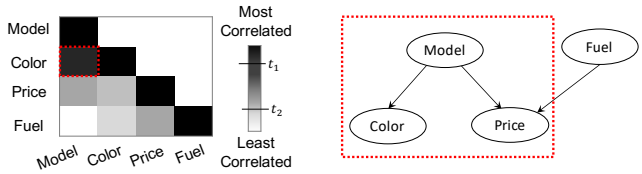
A Statistical Constraint (SC), also known as Probabilistic Dependence and Independence, is a fundamental concept in Machine Learning (ML) [24, 48]. An SC expresses the dependence and independence relationships among a set of variables, and plays an important role in statistical modeling and causal analysis [56]. For example, suppose that a data scientist wants to build a regression model to predict Car Price. She needs to first understand the (in)dependence relationship between each feature (e.g., Color, Model, Fuel) and the target variable (i.e., Price). Many SCs may be relevant, for instance

$$\mathrm{SC}_1 : \mathrm{RowID} \perp\!\!\!\perp \mathrm{Price}, \quad \mathrm{SC}_2 : \mathrm{Model} \not\perp\!\!\!\perp \mathrm{Price}.$$

SC_1 represents an independence relationship, which indicates that RowID cannot be used to predict Price. SC_2 represents a dependence relationship, which indicates that Model is a good feature for predicting Price.

The novel problem we study in this paper is *how to leverage SCs for error detection.* Although SCs have been widely examined and applied, to the best of our knowledge, they have *not* previously been deployed for error detection. The main purpose of this paper is to i) introduce SCs and associated techniques to data cleaning and ii) study how to apply SCs to error detection. In the following, we discuss two application scenarios of SC-based Error Detection for ML.

ML Model Construction. During ML model construction, data scientists may discover a *counter-intuitive* SC from the data, and want to know whether it is a result of data errors.



(a) Correlation Matrix (b) Bayesian Network

Figure 1: (a) A counter-intuitive SC: Color $\perp\!\!\!\perp$ Model (highlighted) is inferred from the correlation matrix; (b) A counter-intuitive SC: Color $\perp\!\!\!\perp$ Price | Model is derived from the bayesian network.

*Work is done while visiting Simon Fraser University

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

ACM ISBN 978-1-4503-6735-6/20/06…$15.00
https://doi.org/10.1145/3318464.3380568

Example 1. *Consider a car database with Model, Color, Price, Fuel attributes. A data scientist computes a correlation matrix from the data (Figure 1(a)), where each cell in the matrix contains the Kendall Tau correlation between two attributes [1] (see Sec. 4.3). She looks for highly correlated cells (dark in the heatmap) and checks whether they contradict her domain knowledge. The data correlation matrix suggests that Color and Model are highly correlated (red dotted line). This is a counter-intuitive relationship since knowing Color should not give significant information about Model. The data scientist suspects that the data may have some errors and wants to confirm whether it is the case.*

Next, she builds a Bayesian network model from the data [45] (Figure 1(b)), where each node represents an attribute and each edge represents a conditional dependency. She can infer some simple SCs from the edges in the model, such as Model $\not\perp\!\!\!\perp$ Color and Model $\not\perp\!\!\!\perp$ Price. By applying d-separation [25], she can infer many more complex SCs from the model, such as Color $\perp\!\!\!\perp$ Price | Model: knowing Color gives no extra information about Price once we have knowledge of Model. If this contradicts her domain knowledge, again she wants to examine whether the data has errors that cause this counter-intuitive independence.

ML Model Deployment. During model deployment, data scientists may infer a number of important SCs from an accepted model, and want to ensure that new data satisfy them.

Example 2. *After a model is trained, it will be applied to unseen test data. If training data and test data satisfy different dependencies (e.g., Model and Fuel are dependent on training data but not on test data), the model performance will degrade. Enforcing SCs on test data captures this* SC violation. *In our experiments, we use a real-world dataset to demonstrate that there data errors can cause an SC to hold on training data but not on test data. Our methods can be applied to detect such SC violations.*

In a similar scenario, training data needs to be collected and updated periodically. Before training a model on incoming datasets, it is important to check whether the updated data satisfies a set of user-specified SCs to ensure the model is trained on a trustworthy data source. For example, suppose that a user specifies that Model and Fuel are dependent, but the new training dataset shows an independence between Model and Fuel. The analyst will want to detect this violation and understand why it happens.

To support such applications, we develop SCODED, an SC-Oriented Data Error Detection system. SCODED contains two major components: *SC Violation Detection* and *Error Drill Down*. Given a dataset and an SC (e.g., Color $\perp\!\!\!\perp$ Model), SCODED first checks whether the dataset violates the SC, i.e., whether the dataset shows a different statistical relationship (Color $\not\perp\!\!\!\perp$ Model). If the violation is detected, it returns the top-k records that contribute most to the violation, aiming to help the user understand why the violation happens.

SC Violation Detection. We find that SCs in many datasets do not hold exactly. Indeed [62] reports that in many datasets, ICs, such as functional dependencies, do not hold exactly either, causing low precision (many false positives). The solution in previous error detection systems is to use *approximate* constraints. Inspired by approximate functional dependencies [44], we define approximate statistical constraints, which hold to a degree and allow for exceptions. We develop a new framework that leverages statistical hypothesis testing to detect the violations of approximate SCs. We also evaluate different hypothesis test statistics for error detection: For categorical (discrete) attributes, the G-test; for continuous attributes, Kendall's rank correlation coefficient τ.

Error Drill Down. Once an SC is detected, a user may ask why it is violated. SCODED helps the user identify likely dirty records that cause the violation. The problem of identifying a subset of dirty records can be phrased as a *dataset partition problem*: find the minimum number of records such that if they were removed from the data, the violation would be resolved. A related problem is the *top-k contribution problem*: find the top-k records that contribute most to the violation. Both problems are novel optimization problems relevant for error detection. We prove that the dataset partition problem reduces to the top-k problem, in the sense that a polynomial-time algorithm for dataset partition yields a polynomial-time algorithm for top-k. We propose an efficient top-k error detection algorithm, whereas the brute-force solution to the top-k problem is highly inefficient.

To place SCODED in the context of previous work, we note that a recent survey paper classifies existing approaches into four categories: constraint-based, pattern-based, outlier detection, and de-duplication [5]. Our paper belongs to the first category, where a user specifies a constraint, and the system detects erroneous values that violate the constraint.

Statistical vs. Integrity Constraints. Various forms of Integrity Constraints (ICs) are proposed for error detection [31]. One natural question is how different SCs are from ICs. Exploring the answer to this question is useful for two reasons. First, since ICs are a well-known concept in the DB community, it will facilitate the understanding of SCs. Second, it will identify the situations where SCs are more expressive than ICs. Previous work established several entailment relationships [22, 34, 55] with functional dependencies (FDs), and multi-valued dependencies (MVD). However, they only apply to *saturated* SCs, where the SC involves *all* columns in a relation. We extend their work for *general* SCs and describe new entailment relationships.

We implement the SC violation detection and top-k error detection algorithms in SCODED. Extensive experiments on synthetic and real-world datasets show the advantages of SCODED over state-of-the-art approaches when detecting errors that cause violations of SCs. Our main contributions are as follows.

- Introducing a novel SC-oriented error detection approach, and identifying scenarios in which SCs are useful and complement existing approaches.
- Leveraging and developing statistical hypothesis testing methods to assess whether a dataset violates an SC, and to what degree the SC holds or fails.
- Extending statistical hypothesis testing methods with a novel top-k approach to highlight records that contribute the most to the violation of statistical constraints. Drilling down on the causes of SC violations identifies records that are likely to be dirty.

The remainder of this paper is organized as follows. We formally define SCs and compare SCs and ICs in Section 2. Section 3 presents the SCODED system architecture. We study the SC violation detection problem in Section 4, and the error-drill-down problem in Section 5. Section 6 reports our experimental findings. We review related work in Section 7, and conclude in Section 8.

2 STATISTICAL CONSTRAINTS

We first present a formal definition of SCs, and then discuss the entailments between SCs and ICs.

2.1 Definitions

A *variable* X is an attribute or feature that can be assigned a value x from a fixed domain; we write $X = x$ to denote an assignment of a value to a variable. Boldface vector notation refers to finite sets of objects. For example $\mathbf{X} = \mathbf{x} \equiv (X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n)$ denotes a *joint assignment* where variable X_i is assigned value x_i, for each $i = 1, \ldots, n$. In relational terms, a variable corresponds to an attribute or column, and a joint assignment to a tuple or row.

A *random variable* requires a distribution $P(X = x)$ that assigns a probability to each domain value in the domain of X. A *joint distribution* $P(\mathbf{X} = \mathbf{x})$ assigns a probability to each joint assignment. Given a joint distribution for a set of variables $\mathbf{X}$, the *marginal distribution* over a subset $\mathbf{Y}$ is defined by $P(\mathbf{Y} = \mathbf{y}) \equiv \sum_{\mathbf{z}} P(\mathbf{Y} = \mathbf{y}, \mathbf{Z} = \mathbf{z})$. Here $\mathbf{Z} = \mathbf{X} - \mathbf{Y}$ contains the set of variables in $\mathbf{X}$ but not in $\mathbf{Y}$, and the comma notation $\mathbf{Y} = \mathbf{y}, \mathbf{Z} = \mathbf{z}$ denotes the conjunction of two joint assignments. The *conditional probability* of an assignment $\mathbf{X} = \mathbf{x}$ given another assignment $P(\mathbf{Y}) = \mathbf{y}$ is defined as $P(\mathbf{X} = \mathbf{x} | \mathbf{Y} = \mathbf{y}) \equiv P(\mathbf{X} = \mathbf{x}, \mathbf{Y} = \mathbf{y}) / P(\mathbf{Y} = \mathbf{y})$.

A key notion of this paper is the concept of *conditional independence among sets of variables.* Intuitively, a set of variables $\mathbf{X}$ is independent of another set $\mathbf{Y}$ given a third conditioning set $\mathbf{Z}$ if knowing the values of the variables in $\mathbf{Y}$ adds no information about the values of the variables in $\mathbf{X}$, beyond what can be inferred from the values in the set $\mathbf{Z}$. Formally, for three disjoint sets $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ and assuming $P(\mathbf{Z} = \mathbf{z}) > 0$ for all values $\mathbf{z}$, we define

$$\begin{aligned} \mathbf{X} \perp\!\!\!\perp \mathbf{Y} | \mathbf{Z} \quad &\equiv \\ P(\mathbf{X} = \mathbf{x}, \mathbf{Y} = \mathbf{y} | \mathbf{Z} = \mathbf{z}) \quad &= \\ P(\mathbf{X} = \mathbf{x} | \mathbf{Z} = \mathbf{z}) \times P(\mathbf{Y} = \mathbf{y} | \mathbf{Z} = \mathbf{z}) \quad &\text{for all } \mathbf{x}, \mathbf{y}, \mathbf{z}. \end{aligned}$$

We call $\mathbf{X} \perp\!\!\!\perp \mathbf{Y} | \mathbf{Z}$ an *independence SC (ISC).* A *dependence SC (DSC)*, written $\mathbf{X} \not\perp\!\!\!\perp \mathbf{Y} | \mathbf{Z}$, is the negation of an ISC: for *some* values $\mathbf{x}, \mathbf{y}, \mathbf{z}$, we have $P(\mathbf{Y} = \mathbf{y}, \mathbf{Z} = \mathbf{z}) > 0$ and $P(\mathbf{X} = \mathbf{x} | \mathbf{Y} = \mathbf{y}, \mathbf{Z} = \mathbf{z}) \neq P(\mathbf{X} = \mathbf{x} | \mathbf{Z} = \mathbf{z})$.

Definition 1 (Statistical Constraints). *Fix a set of variables $\mathbf{V} = \{V_1, \ldots, V_n\}$. A finite set of statistical constraints $\Sigma = \mathcal{I} \cup \mathcal{D}$ comprises*

(1) a finite set of independence SCs, $\mathcal{I} = \{\phi_1, \ldots, \phi_p\}$, where each ϕ_i is of the form $\mathbf{X} \perp\!\!\!\perp \mathbf{Y} | \mathbf{Z}$

(2) a finite set of dependence SCs, $\mathcal{D} = \{\lambda_1, \ldots, \lambda_q\}$, where each λ_i is of the form $\mathbf{X} \not\perp\!\!\!\perp \mathbf{Y} | \mathbf{Z}$

So far we have defined SCs for probability distributions in general. They become constraints on data when we apply them to the observed probability distribution associated with a data relation [34]. This **empirical distribution** P_D associated with a relation D is defined as follows. Let $r[\mathbf{X}]$ denote the tuple of values in the $\mathbf{X}$ columns for record r. Given an assignment $\mathbf{X} = \mathbf{x}$, a record *satisfies the assignment* if $r[\mathbf{X}] = \mathbf{x}$. The **empirical count** is the number $N_D(\mathbf{X} = \mathbf{x})$ of records that satisfy it. The **empirical frequency** of an assignment is the number of satisfying records, divided by the total number of records:

$$P_D(\mathbf{X} = \mathbf{x}) \equiv N_D(\mathbf{X} = \mathbf{x}) / N_D$$

where N_D is the cardinality of relation D.

Table 1: Entailments Between SCs and ICs.

ISC vs. FD	$Y \perp\!\!\!\perp (X \cup Y)^C \mid X$	$\Longleftarrow$	$X \rightarrow Y$
ISC vs. MVD	$Y \perp\!\!\!\perp (X \cup Y)^C \mid X$	$\Longleftrightarrow$	$X \quad Y$
ISC vs. EMVD	$Y \perp\!\!\!\perp Z \mid X$	$\Longrightarrow$	$X \quad Y \mid Z$
DSC vs. FD	$Y \not\perp\!\!\!\perp X$ DSC is maximally strong	$\Longleftarrow$	$X \rightarrow Y$

2.2 Statistical vs. Integrity Constraints

ICs that represent *multi-column dependencies among attributes*, such as functional dependencies and embedded multi-valued dependencies, are conceptually similar to SCs: Both represent *inferential relevance* from values in one set of columns to values in another. This conceptual similarity leads to formal

Table 2: The table satisfies Z $X|Y$, but not $X \not\perp\!\!\!\perp Y|Z$. Counter-example to the converse of Proposition 1.

	Z	X	Y	M
r_1	z_1	x_1	y_1	m_1
r_2	z_1	x_2	y_2	m_1
r_3	z_1	x_1	y_2	m_1
r_4	z_1	x_1	y_2	m_2
r_5	z_1	x_1	y_2	m_3
r_6	z_1	x_2	y_1	m_1

connections between SCs and FDs/EMVDs. The motivation for these results is to provide a theoretical understanding of the differences between ICs and SCs, especially when one is applicable for error detection, and the other is not. Table 1 summarizes the theoretical relationships among ICs and SCs discussed in this section.

We review theorems established previously in the literature [34] and add two new results. i) An FD entails a maximally strong dependence SC (DSC). ii) An independence SC (ISC) entails an EMVD. The results are used as follows in our experiments in Section 6. i) means that we can translate an FD into a DSC and use SCODED as a novel method for leveraging an FD for error detection. Our experiments show that this is an effective error detection method given an approximate FD. ii) shows the novelty of ISCs for error detection: While EMVDs have not been previously used in error detection [6], we can think of ISCs as a way to leverage the inferential independence expressed by EMVDs. Our experiments show that using an approximate ISC as an approximate EMVD allows us to detect new kinds of errors, compared to deterministic dependencies (such as functional dependencies and dependence constraints).

We say that one constraint entails another, denoted by $\Rightarrow$, if any relation D that satisfies the former also satisfies the latter. A relation D satisfies a statistical constraint if the empirical data distribution P_D does [34]. Two common ICs that we consider in this section are functional and embedded multi-valued dependencies, which are defined as follows.

Definition 2 (Functional Dependency (FD)). *A relation D satisfies an FD $X \to Y$ if for any two records $r_1, r_2 \in D$, if $r_1[X] = r_2[X]$, then $r_1[Y] = r_2[Y]$.*

For example, consider an FD $Z \to X$. Table 2 does not satisfy this FD since for r_1 and r_2, we have $r_1[Z] = r_2[Z] = z_1$, but $r_1[X] = x_1$ and $r_2[X] = x_2$ are not equal.

Definition 3 (Embedded Multi-valued Dependency (EMVD)). *A relation R satisfies an EMVD X $Y|Z$ if $\Pi_{XYZ}(D) = \Pi_{XY}(D) \bowtie \Pi_{XZ}(D)$ [34].*

For example, consider an EMVD: Z $X|Y$. Table 2 satisfies this EMVD. The reason is as follows. From the table we have $dom(Z) = \{z_1\}$, $dom(X) = \{x_1, x_2\}$ and $dom(Y) = \{y_1, y_2\}$. Given that $Z = z_1$, the columns B and C contain all combinations of their domain values. Immediately, we have $\Pi_{ZXY}(D) = \Pi_{ZX}(D) \bowtie \Pi_{ZY}(D)$.

A multi-valued dependency (MVD) X Y is the special case of a *saturated* EMVD X $Y|Z$ where $Z = (X \cup Y)^C$, that is, XYZ contains all columns in the relation.

Independence Statistical Constraints(ISC). Key entailments established in previous work [34] are the following.

$$X \to Y \implies X \quad Y \iff Y \perp\!\!\!\perp (X \cup Y)^C | X$$

It is well-known that an FD $X \to Y$ entails an MVD X Y [22]. The equivalence between an MVD X Y and a saturated ISC $Y \perp\!\!\!\perp (X \cup Y)^C | X$ was established by [55].

Since MVD is a special case of EMVD, and a saturated SC is a special case of SC, we explore the entailment relationship between their general forms, i.e., EMVD vs. SC. Interestingly, EMVDs are weaker constraints than SCs. That is, an ISC entails an EMVD, but not vice versa.

Proposition 1. *For all disjoint sets of attributes X, Y, Z*

$$Y \perp\!\!\!\perp Z|X \implies X \quad Y|Z \tag{1}$$

Proof. All the proofs in the paper can be found in the technical report [69].

Table 2 shows a counter example. As shown above, Table 2 satisfies Z $X|Y$. However, $P(X = x_1|Z = z_1) = \frac{2}{3}$, $P(Y = y_1|Z = z_1) = \frac{1}{3}$, and $P(X = x_1, Y = y_1|Z = z_1) = \frac{1}{6}$, which leads to $P(X, Y|Z)$, $P(X|Z)P(Y|Z)$, so $X \not\perp\!\!\!\perp Y|Z$.

Proposition 1 shows the novelty of ISCs for error detection: approximate ISCs can be seen as a kind of approximate EMVD, and our paper is the first to leverage this kind of constraint for error detection.

Dependence Statistical Constraints (DSC). A common measure of dependence strength is the **mutual information** $I(X;Y)$ [34, 65]:

$$I(X;Y) \equiv \tilde{O}_{x;y} P(X = x, Y = y) \log_2\left(\frac{P(Y = y, X = x)}{P(Y = y) \times P(Y = y)}\right).$$

Two categorical random variables have minimal mutual information 0 if and only if they are independent [34]. Let I_D be the mutual information derived from the empirical data distribution P_D. An FD entails a maximally strong DSC:

Proposition 2. *Suppose that a relation D satisfies a functional dependency $X \to Y$. Then P_D satisfies the DSC $X \not\perp\!\!\!\perp Y$ of MI-maximal strength: $I_D(X;Y) \geq I_D(X';Y)$ for all sets of columns X'.*

The proposition assumes that the Y-columns are not constant (contain only a single repeated tuple). Similar results can

(a) Original Car Database

RID	Model	Color
r1	BMW X1	White
r2	BMW X1	Black
r3	BMW X1	White
r4	BMW X1	Black
r5	Toyota Prius	White
r6	Toyota Prius	White
r7	Toyota Prius	White
r8	Toyota Prius	Black

(b) Inserted New Records

RID	Model	Color
r9	BMW X1	White
r10	BMW X1	White
r11	BMW X1	White
r12	BMW X1	Black
r13	Toyota Prius	Black
r14	Toyota Prius	Black
r15	Toyota Prius	Black
r16	Toyota Prius	Black

Figure 2: (a) The original Car database and Model ⫫ Color. (b) Adding records $r9$–$r16$ to the database. SCODED first detects that the new database violates Model ⫫ Color, and then finds the top-5 records (highlighted), which potentially cause the violation.

be shown for continuous variables. Proposition 2 supports translating an FD $X \rightarrow Y$ into a DSC $X \not\perp\!\!\!\perp Y$. This is the translation that we use in our experiments in Section 6.

3 SCODED OVERVIEW

We describe SCODED, an SC-Oriented Data Error Detection System. Figure 3 shows the architecture. The system consists of four core components: *SC Discovery*, *Consistency Checking*, *Violation Detection* and *Error Drill-down*.

SC Discovery. This component aims to assist data scientists to discover SCs. Conceptually it is similar to discovering dependency ICs, which is an important part of the *data profiling process* [6]. Various statistical tools have been developed that a data scientist can leverage to perform *statistical data profiling* and discover SCs (cf. Figure 1). Another approach used very recently in data management [58, 59] is to visualize the causal relationships among random variables in a *causal graph*. From a causal graph all possible SCs can be derived. SC discovery is a major topic in the AI community [16, 24, 48]. Given the extensive previous research in both statistics and machine learning on how to solicit SCs from users, we include SC Discovery as a component, but do not propose a new method for it.

Consistency Checking. This component aims to check whether a set of SCs have conflicts. Consider a simple example, where we are given two SCs: $\Sigma = \{X \perp\!\!\!\perp Y, X \not\perp\!\!\!\perp Y\}$. Clearly, $X \perp\!\!\!\perp Y$ and $X \not\perp\!\!\!\perp Y$ cannot be satisfied by the same data relation, so Σ is inconsistent. New potentially contradictory SCs can be derived using the graphoid axioms [50]. Fortunately, the problem of deciding whether a given set of SCs is consistent has been studied extensively by AI researchers [24, 48, 61]. Given the extensive previous research conducted on SC consistency checking, we include it as a component, but do not propose a new method for it.

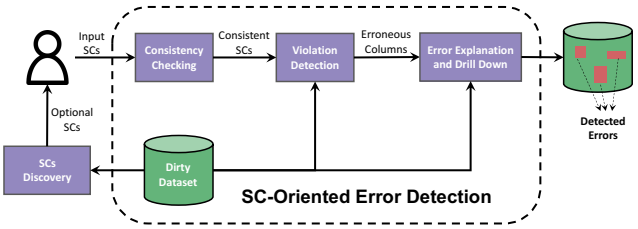


Figure 3: SCODED Architecture

Violation Detection. This component aims to check whether a dataset violates an SC by deploying hypothesis testing. A hypothesis test can be used to determine whether the violation is statistically significant or potentially due to random exceptions. For example, consider the original car database and SC = Model ⫫ Color in Figure 2(a). Suppose the database is updated by inserting records $r_9 - r_{16}$ (Figure 2(b)). The user expects that Model and Color should have an independent relationship. However, a G hypothesis test on the data indicates that Model and Color are highly correlated, which violates the constraint.

Error Drill-Down. If a dataset violates an SC, the data scientist will want to investigate the cause of the violation. This error-drill-down component aims to identify individual records that contribute the most to the violation of an SC. In many situations, systematic errors cause the SC violation as a side-effect. Drill-down analysis shows the user a small number of k potentially dirty records. The user can check whether these records follow a pattern. Continuing the example in Figure 2, if the user specifies $k = 5$, the error-drill-down algorithm will return $r_8, r_{13} - r_{16}$. The data scientist finds two interesting patterns: (1) all five records are from Toyota Prius, and (2) their colors are all Black.

4 DETECTING SC VIOLATIONS

We present an SC violation detection framework that leverages statistical hypothesis testing. *A hypothesis test statistic can be viewed as an approximate SC.* The motivation for using approximate SCs is that in many datasets, SCs do not hold exactly. Inspired by approximate functional dependencies [44], we define approximate statistical constraints, which hold to a degree and allow for exceptions. We investigate which test statistics are appropriate for error detection.

4.1 Approximate SCs

Consider the Car database. Suppose a data scientist wants to represent the domain knowledge that Price and Color have a *small* dependence. We find that neither Price ⫫ Color nor Price ⫫̸ Color meets her need since the former represents an independence (rather than dependence) relationship while the latter does not characterize "small" in the dependence

Algorithm 1: SCs Violation Detection Algorithm

Input: A dataset D, an approximate SC $\langle \phi_{X \perp\!\!\!\perp Y|Z}; \alpha \rangle$
Output: Violated or Not Violated

```
// Apply Hypothesis Testing
Let c = φ_{X ⫫ Y|Z}(D)
Compute p-value p(D) = P(t ≥ c | X ⫫ Y|Z)
if p(D) < α then
    Reject X ⫫ Y|Z
    return Violated
else
    Accept X ⫫ Y|Z
    return Not Violated
```

correlation. Approximate SCs address this problem, because they allow us to define a small degree of dependence as one that is close to minimal, or 0; similarly a strong degree dependence is one that is close to maximal.

For ease of presentation, we will focus on independence SCs but note that all the definitions and algorithms proposed in Sections 4 and 5 can be trivially extended to dependence SCs. Let $\phi_{X \perp\!\!\!\perp Y|Z}$ denote a test statistic (examples appear in Section 4.3). A test statistic is an aggregate function that, given a relation, returns a real number in $[0, m]$, where the value 0 corresponds to minimal dependence or complete independence, while the value m to minimal independence or maximal, typically deterministic, dependence. *The larger the statistic $\phi_{X \perp\!\!\!\perp Y|Z}(D)$, the stronger the dependence.* Let α denote a false dependence rate (FDR), which controls the approximation level. Our FDR concept is an instance of the general concept of a *significance level.*

Definition 4 (Approximate Statistical Constraints). *Define an approximate SC as $\langle \phi_{X \perp\!\!\!\perp Y|Z}, \alpha \rangle$, where $\phi_{X \perp\!\!\!\perp Y|Z}$ is a test statistic and $\alpha \in [0, 1]$ is a false dependence rate.*

Higher FDRs require stronger independence relationships. A maximal FDR of 1.0 corresponds to complete independence, or a statistic value 0. For example, $\langle \phi_{\text{Model} \perp\!\!\!\perp \text{Color}}, 0.90 \rangle$ enforces a strong independence between Model and Color. A minimal FDR of 0.0 corresponds to a maximally high value of the test statistic. For example, $\langle \phi_{\text{Model} \perp\!\!\!\perp \text{Price}}, 0.05 \rangle$ enforces a weak independence between Model and Price.

4.2 SC Violation Detection Algorithm

For simplicity, we first focus on the case when X and Y are single variables, and then extend the framework to the general setting.

X, Y are single variables. A *hypothesis test* T is a procedure that takes as input a dataset D and outputs either 0 ("the hypothesis is rejected") or 1 ("the hypothesis is not rejected"). The most common hypothesis tests are based on the concept of a *p-value*, which is the probability of observing a value t at least as great as the test statistic for the dataset, if the *null hypothesis = independence constraint* were true. Formally, the p-value for a dataset is computed as $p(D) = P(t > c \mid H_0 \text{ is } true)$ where H_0 is the NULL Hypothesis, c is the standard test statistic value calculated from the data set, and $t > c$ defines the interval of real numbers greater than c . *The higher the p-value, the lower the observed test statistic, the stronger the observed independence, and the weaker the observed dependence.* For example, a maximal p-value of 1.0 corresponds to a test statistic value of 0 and hence to maximally strong independence. A minimal p-value of 0.0 corresponds to a maximally high test statistic and hence to a maximally strong dependence. The test rejects H_0 if the observed p-value is less than the specified FDR α. This means that the significance level α is an upper bound on the probability of a false positive, that is, rejecting the null hypothesis when it actually is true. Definition 5 formally defines independence SC violation.

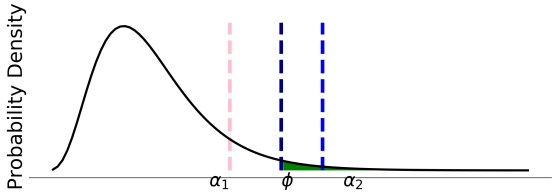


Figure 4: The x-axis represents values ϕ of the test statistic. We project two different FDR rates α_1 and α_2 to their corresponding ϕ-values.

Definition 5 (SC Violation). *Given a dataset D, and an approximate SC: $\langle \phi_{X \perp\!\!\!\perp Y|Z}, \alpha \rangle$, we say the dataset violates the given SC if and only if*

$$P(t > c \mid H_0) < \alpha \tag{2}$$

where $c = \phi_{X \perp\!\!\!\perp Y|Z}(D)$, and $H_0 : X \perp\!\!\!\perp Y|Z$.

Algorithm 1 illustrates the violation detection algorithm.

Example 3. *Figure 4 illustrates the density probability P_D over the Car dataset D. Consider an approximate SC: $\langle \phi_{Model \perp\!\!\!\perp Price}(\cdot), 0.05 \rangle$. The shaded green region corresponds to $p(D)$. We project the p-value*

$$p(D) = P(t > \phi_{Model \perp\!\!\!\perp Color}(D) \mid H_0 \equiv \ Model \perp\!\!\!\perp Color)$$

on the x-axis as ϕ (refer to the navy dashed line). Consider two FDRs α_1 and α_2. If α_1 is chosen, then we have $p(D) > \alpha_1$. Hence Algorithm 1 will accept H_0 and return "not violated". However, if α_2 is chosen, then we have $p(D) < \alpha_2$. Algorithm 1 will reject H_0 and return "violated".

X, Y are sets of variables. Our violation detection algorithm can be easily extended to the case when X, Y are sets of variables. Consider an SC: $X \perp\!\!\!\perp Y_1 Y_2 | Z$, where X, Y_1, Y_2, and Z are sets of variables. The *decomposition principle* asserts that this SC can be equivalently represented by the

following two SCs [24]:

$$X \perp\!\!\!\perp Y_1Y_2|Z \iff (X \perp\!\!\!\perp Y_1 \mid ZY_2) \,\&\, (X \perp\!\!\!\perp Y_2 \mid ZY_1).$$

We apply decomposition recursively until there is no SC whose X or Y contains multiple variables. Then, the violation detection problem is reduced to the single-variable case. Decomposition can be derived from mild theoretical assumptions, and has been empirically verified on many datasets [11, 48].

4.3 Choosing Test Statistics

SCODED adopts the G-test for categorical variables and Kendall's τ rank for numerical variables.

G-test. We use the G statistic for testing independence among categorical (discrete) variables. G is the rescaled mutual information for the empirical distribution P_D:

$$G(D) \equiv 2 \cdot N_D \cdot I(X;Y).$$

Rescaling by the size of the data relation N_D is important because a dependency based on a larger dataset is more likely to indicate a genuine error rather than random fluctuations.

Kendall's τ rank. We use *Rank Correlations* for testing independence among numerical (continuous) variables. Consider n datapoints with two features $D = (x_1, y_1), \ldots, (x_n, y_n)$. For two datapoints (x_i, y_i) and (x_j, y_j) with $i \neq j$, if $x_i > x_j$ and $y_i > y_j$, or $x_i < x_j$ and $y_i < y_j$, then the two variables agree on the ordering of i and j and the pair (i, j) is *concordant*. The number of concordant pairs is denoted as $n_c(D)$. Else if $x_i > x_j$ and $y_i < y_j$, or $x_i < x_j$ and $y_i > y_j$, then the two variables disagree on the ordering of i and j and the pair (i, j) is *discordant*. The number of discordant pairs is denoted as $n_d(D)$. Pairs neither concordant nor discordant are called *tied*, and their number is denoted as $n_t(D)$. The τ statistic is then computed as $\tau(D) \equiv \frac{n_c(D) - n_d(D)}{\binom{n}{2}}$.

Motivation. We chose Kendall's τ as a default method over other popular options (e.g., Pearson's coefficient ρ and Spearman's ρ_S [65]) because the default method in SCODED should be compatible with many data characteristics, so the *fewer underlying data assumptions*, the better. In statistical terminology, we want to use a *non-parametric* hypothesis test. Pearson's ρ is a *parametric* method that measures the degree to which X and Y are linearly related. The disadvantage of the ρ statistic is that it is reliable only under certain assumptions, including that the relationship between X and Y should be approximately linear. Spearman's ρ_S computes the linear correlation ρ for the ranks of data points from each column. Comparison studies [17, 23, 29, 36, 67] have found that Kendall's τ is generally more robust in avoiding false positives, which makes it preferred for data error detection.

Computing p-values. The most common approach to computing p-values (Equation (2)) is to approximate the distribution of the test statistic by a reference distribution (χ^2 for G and Gaussian for τ) [65]. This allows p-values to be computed in closed form. Within the sample size limit, the p-value approximations are exact under the common assumption that the data points are i.i.d. (independently and identically distributed). For a finite sample size, the traditional rule is that the χ^2 approximation to the G-test is close enough if each of the expected counts is at least 5 (in symbols, $N_D(X = x) \times N_D(Y = y)/N_D \geq 5$).[1] For the τ-test, the Gaussian approximation is sufficiently close for $N_D > 60$.[2] In the case of conditional tests, the sample size $N_D(Z = z)$ must be sufficiently large for each value of the conditioning variables Z. If the sample size is too small to use the closed-form approximation, p-values can be computed using what is known as an exact test [65]. Exact tests are computationally more costly but fast for small sample sizes.

5 ERROR DRILL DOWN

This section aims to develop efficient algorithms to identify the top-k records that contribute the most to a detected violation. We propose a general framework and formalize optimization problems in Section 5.1, then present two search strategies in Section 5.2, and finally describe efficient top-k error detection algorithms in Section 5.3.

5.1 Explanation as Optimization

An SC violation helps a user detect which columns have errors, but not which rows. To this end, we develop an interactive error-drill-down framework. A user specifies an SC and a hypothesis testing method (e.g., τ test). If the result of the hypothesis test indicates that a data error may exist, the drill-down method will return k records whose column values are most likely to cause the violation. We cast error drill down as two optimization problems: dataset partition and top-k contribution. These are algorithmically equivalent (mutual polynomial-time reductions exist).

A straightforward approach is to assume that data violating an SC can be partitioned into a clean and a dirty subset. The clean subset satisfies the SC, whereas the dirty one does not. Therefore removing the dirty subset will bring the value of the test statistic closer to what the SC requires. This leads to the following optimization problem.

Definition 6 (Dataset Partition). *Given a dataset D, and an approximate independence SC $\langle \phi_{X \perp\!\!\!\perp Y \mid Z}, \alpha \rangle$, the data partition problem is to find a minimum-cardinality subset ΔD of records such that the updated p-value $p(D - \Delta D) > \alpha$.*

[1] http://www.biostathandbook.com/small.html

[2] https://itl.nist.gov/div898/software/dataplot/refman1/auxillar/kend_tau.htm

Another approach is to search for a set of records that minimize the test statistic.

Definition 7 (Top-k Contribution). *Given a dataset D, an SC: $X \perp\!\!\!\perp Y|Z$, a test statistic ϕ, and a threshold k, the top-k contribution problem identifies a set of k records ΔD, such that the test statistic $\phi_{X \perp\!\!\!\perp Y|Z}(D - \Delta D)$ is minimized.*

The top-k contribution problem asks the user to provide a threshold k rather than a FDR α. Theorem 1 asserts that a tractable algorithm for solving one of the optimization problems leads to a tractable algorithm for the other.

Theorem 1. *The dataset partition and the top k-problem can be reduced to each other in polynomial time.*

We focus on addressing the top-k contribution problem in the following. The main reason is that a threshold k is even more intuitive to the user than the significance level α. Also, the performance of other error detection methods can be compared with SCODED using metrics like recall@k, because these methods support retrieving the top-k records labelled as potential errors. But not all error detection methods support specifying a bound α on the rate of false positives.

5.2 Search Strategies

A brute-force solution is to enumerate all $\binom{|D|}{k}$ possibilities, and then return the best result. This is prohibitively expensive. Even with a modest data size of 10,000 records, assuming a reasonable k = 10, this approach would require enumerating $\binom{10000}{10} = 2.7 \times 10^{33}$ possibilities. Therefore, we propose two greedy algorithms, *K strategy* and *K^c strategy*. The K strategy seeks to directly identify the best k records; the K^c strategy seeks to remove the worst $n - k$ records and then return the remaining k records as a result.

K Strategy. The algorithm first selects the *best-to-remove* record d^* from D such that if it was removed, the statistic improves the most. The algorithm removes d^* from D, and then repeats the above process to select the best record from $D - \{d^*\}$. After k iterations, the top-k records are identified.

K^c Strategy. The algorithm first selects the *worst-to-remove* record d' from D such that if it was removed, the statistic would deteriorate the most. The algorithm removes d' from D, and then repeats the above process to select the worst-to-remove record from $D - \{d'\}$. After $n - k$ iterations, where $n = |D|$, the remaining unselected k records are returned.

Remark. The K strategy is more efficient than the K^c strategy because the former only needs to select k records but the latter needs to check $n - k$ records. In terms of effectiveness, the K strategy often leads to a better objective value (i.e., smaller $p(D - \Delta D)$ because it directly optimizes for that value. In our experiments we found that the K^c strategy is particularly useful in identifying a set of k records that are highly correlated with each other, thus it is more suitable to detect errors for the violation of an independence SC.

	BMW X1	Toyota Prius
White	192	200
Black	213	189
Blue	54	369

Figure 5: Group counts of Model and Color

5.3 Top-k Error Detection Algorithms

We describe top-k error detection methods for the two statistics that we examine in this paper, G and τ. We discuss how to implement the K strategy for them efficiently. The same implementation can be extended to the K^c strategy trivially.

Categorical Data. We use the G-test as the default method to detect errors for categorical data. If two records have the same values on the tested columns, there is no difference in choosing either one of them. Therefore, we can group the records based on the two columns that appear in the SC, and reduce the computation by selecting, at each iteration, a group rather than a record. Each group contributes a single term g to the total G-statistic. We randomly pick one record from the group with the highest g-value.

For example, consider an SC: Model $\perp\!\!\!\perp$ Color and a dataset similar to Figure 2 but with more records. We first group the records based on the tested columns (i.e., Model and Color). Suppose there are 2 car models and 3 colors. Then, there will be 6 groups in total. Figure 5 illustrates the 6 groups, where the number in each cell represents the total number of the records that belong to that group. Each cell has a G value z.

Numerical Data. We discuss how we employ the framework for numerical data with the τ test. For investigating an independence SC, the top-k problem (Defn. 7) is as follows.

Definition 8 (top-k for τ-test). *Given a dataset D, an independence SC, and k, find a subset ΔD of records from D that minimizes the τ statistic:*

$$\underset{\Delta D \subseteq D}{\operatorname{argmin}} \frac{n_c(D - \Delta D) - n_d(D - \Delta D)}{\binom{|D|-k}{2}} \quad s.t. \quad |\Delta D| = k$$

We omit the denominator of the objective function since it is a constant function of n only. The K strategy works as follows. The *weight* of a record pair (r_1, r_2) is 1,-1,0 for concordant, discordant, and tied, respectively. A priority queue ranks records by their *benefit* benefit(r). The benefit for a given record r is calculated as follows: find all the records pairs with r, and then sum the weights of these pairs. At each iteration, we select the record with the largest benefit, and update the benefit of the remaining records.

Efficiency Analysis. Each of the k update steps runs in time linear in the number of data records. Therefore, the

Algorithm 2: Efficient τ-test-based error detection algorithm

Input: An SC $= X \perp\!\!\!\perp (\not\perp) Y$, Dataset $D = \{< x_1; y_1 >; \cdots < x_n; y_n >\}, k$
Output: k records

```
T ← ∅; //Segment Tree
Q ← ∅; // Priority Queue
R ← ∅; // Returned List
benefit(< x_i; y_i >) = 0 for < x_i; y_i > ∈ D ;
Sort D by X column value by ascending order of X;
// Initialization tree T_1
for r_i ∈ D do
    n_c = T_1:query((−∞; y_i));
    n_d = T_1:query((y_i; +∞));
    benefit(< x_i; y_i >) = n_c − n_d;
    T_1:Nodeinsert([y_i; y_i]);
    Q:push(< x_i; y_i >; benefit(< x_i; y_i >);
Sort D by X column value by descending order of X;
// Initialization tree T_2
for r_j ∈ D do
    n_d = T_2:query((−∞; y_j));
    n_c = T_2:query((y_j; +∞));
    benefit(< x_j; y_j >)+ = n_c − n_d;
    T_2:Nodeinsert([y_i; y_i]);
    Q:update(< x_j; y_j >; benefit(< x_j; y_j >);
// Iteration
for i = 1 to k do
    Add Q:top() to R;
    Update Q; // Update the weight of each record benefit(< x_i; y_i >) ∈ Q
      by querying the segment tree T_1; T_2
return R;
```

main computational bottleneck of the K strategy is the initialization phase. Consider a dataset with two columns: $D = \langle x_1, y_1\rangle, \cdots, \langle x_n, y_n\rangle$. Initial benefits can be computed efficiently in $O(n \log n)$ steps with two **segment trees**. A segment tree is a tree data structure, where each node stores information about a segment. It allows for inserting a segment and querying a segment in $O(\log n)$ time. Algorithm 2 shows the pseudo-code and Figure 6 presents an example.

Each tree is initialized to be a single node for the interval $(-\infty, +\infty)$. The ascending segment tree is built up as follows: Sort D by column X ascending, and for each record r_i, run two queries $(y_i, +\infty)$ and $(-\infty, y_i)$ against the current tree and the data. Insert a segment data record. The descending segment tree is built the same way, but with column X in descending order. We need $O(\log n)$ time to process each record, thus the total time complexity for building the trees is $O(n \log n)$. After the two trees have been built, the benefit of each record can be computed in $O(\log n)$ time. With this optimization, error drill-down scales to millions of records.

Example 4 (Constructing Segment Tree). *Figure 6 illustrates constructing the ascending segment tree querying* $(-\infty, y]$. *Given the dataset D and SC: Price* $\not\perp$ *Fuel, sort the dataset by Price ascending. Given* $r1$, *query the segment* $(-\infty, y_1]$ *(Fig. 6 (b)). As there is one overlapping range, update the SUM count and then insert* $(-\infty, y_1]$. *Next, query* $(-\infty, y_2)$, *for the overlapping range, update the total sums at each node (Fig. 6 (c)), and insert the segment* $(-y_1, y_2])$. *For* $r3$, *the segment*

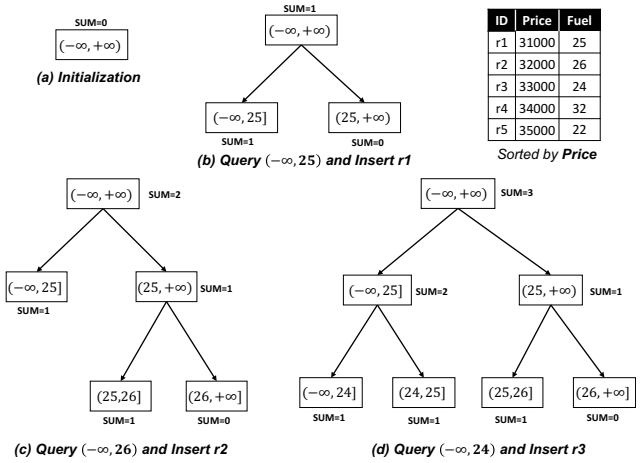


Figure 6: Segment Tree Example

$(-\infty, y_3]$ *overlaps the left-side tree, so we update the total* $r1$ *node sum and the root node sum, and insert* $(-\infty, y_3]$.

6 EXPERIMENTAL EVALUATION

We compare the effectiveness of SCODED with other state-of-the-art error detection approaches, using real-world datasets containing both synthetic and real-world errors.

6.1 Experiment Setup

Datasets. Our evaluation employs six real-world datasets. SENSOR and HOSP are commonly used for evaluating data cleaning algorithms. HOCKEY, CAR, BOSTON, and Nebraska come from the Machine Learning community.

(1) SENSOR[3]. The Sensor dataset comprises the sensor reports from the Berkeley/Intel Lab. The dataset has more than 2 million records, reporting humidity and temperature measurements from 54 different sensors.

(2) HOSP[4]. The HOSP dataset contains 100K records with 19 attributes. We used the same clean and dirty versions of the dataset as previous data-cleaning studies [14, 51].

(3) HOCKEY[5]. The Hockey dataset collected the records of each National Hockey League (NHL) game from 1998-2010. More than ten columns describe player attributes and performance statistics for a season. It was first collected and cleaned for NHL draft prediction [40].

(4) CAR[6]. The Car Evaluation dataset is from UCI Machine Learning repository. This dataset contains seven attributes. We used 4 attributes: Buying price (BP), Car Class (CL), Doors (DR), and Safety level (SA).

[3] http://db.csail.mit.edu/labdata/labdata.html
[4] http://www.hospitalcompare.hhs.gov
[5] https://www.nhl.com
[6] https://archive.ics.uci.edu/ml/datasets/Car+Evaluation.

Table 3: Constraints used by SCODED and other approaches

Attributes	SCODED	Dataset	Integrity Constraints
Temperatures (T) of Neighboring Sensors	$T_a \not\perp\!\!\!\perp T_b$	Sensor	$\forall r_i; r_j \in D : \neg(r_1[T_a] > r_2[T_b] \wedge r_1[T_b] \leq r_2[T_b])$
Rooms(R), Black Index(B)	$R \perp\!\!\!\perp B$	Boston	×
Tax rate, Black Index, Crime(C)	$TX \not\perp\!\!\!\perp B \mid C$	Boston	$\forall r_i; r_j \in D : \neg(r_1[C] = r_2[C] \wedge r_1[T] > r_2[T] \wedge r_1[B] \leq r_2[B])$
N_oxide, Black Index, Tax rate (T)	$N \perp\!\!\!\perp B \mid TX$	Boston	×
Buying Price(BP), Class(Cl)	$BP \not\perp\!\!\!\perp Cl$	Boston	$\forall r_i; r_j \in D : \neg(r_1[BP] > r_2[BP] \wedge r_1[Cl] \leq r_2[Cl])$
Safety(SA), Doors(DR)	$SA \perp\!\!\!\perp DR$	CAR	×
Zip Code(ZIP), City(CY)	$ZIP \not\perp\!\!\!\perp CY$	HOSP	$ZIP \rightarrow CY$ at 25% rate
ZIP, State(ST)	$ZIP \not\perp\!\!\!\perp ST$	HOSP	$ZIP \rightarrow ST$ at 25% rate

(5) `BOSTON`[7] . The Boston dataset was taken from the Boston Standard Metropolitan Statistical Area (SMSA) in 1970. This dataset was first used to study the relationship between clean air quality and households' willingness to pay [26]. There are 506 instances. We used 6 of 14 attributes: Distance to CBD area-Distance (D), Nitric Oxides Concentration-N_oxide (N), Crime Rate-Crime (C), Black index of population (B), Rooms (R) and Tax Rate (Tx).

(6) `Nebraska`[8] . The Nebraska dataset collected the measurements from national weather stations from 1949 to 1999. The dataset was often used to predict weather trends with many features. We followed previous work [8, 21] and selected 8 features from the dataset of the region Bellevue, Nebraska. Eight features include Temperature(TM), Dew Point Temperature(DT), Sea level Pressure(SLP), Station Pressure(STP), Visibility(V), Wind(WD), Maximum Sustained Wind(MW), Maximum Wind Gust(G). The weather situation (WS) is the prediction label.

Real-world Errors. Hockey is a public dataset used for hockey player draft analytics. The creators of the dataset originally believed that the dataset had no error. The analysis, however, found that the downstream machine learning model predicts results that contradict domain knowledge, indicating that the data is not clean.

The Nebraska dataset contains missing values. We followed the previous approaches in [8, 21] and imputed the missing value with the mean value from previous 30-days records. However, in addition to the errors previously documented, we found outliers in some of the columns.

The large Sensor dataset contains millions of temperature measurements by Intel Lab. To compress it, we replaced sensor readings by their hourly average collected in [42]. The dataset is known to contain erroneous outliers [33, 35]. A standard way to pre-process Sensor data is to remove obvious outliers and then impute the missing value [47]. However, such imputations might impact downstream ML models. Therefore, we try to detect the errors that result from removing outliers and imputing their values.

[7]https://www.cs.toronto.edu/ delve/data/boston/bostonDetail.html

[8]https://catalog.data.gov/dataset/global-surface-summary-of-the-day-gsod

Simulated Errors. We simulated two common types of errors: *sorting error* and *imputation error*, both of which have appeared in real-world model development [53]. An imputation errors occur when a missing value is filled in with a misleading value, typically one or more constants. The Hockey dataset errors are due to imputations (see Case Study below). There are two famous machine learning examples of a sorting error occurred in the KDD-Cup 2008, which dealt with cancer detection from mammography data. A sorting error introduced a strong dependence relationship between Patient ID and the class label. A team found this error and utilized the dependence relationship to win the competition [53]. In a real machine learning system, however, Patient ID should not be used to predict whether a patient has cancer or not. A similar dependence based on a sorting error was used to win a Kaggle competition [4].

For synthetic sorting errors, we selected $\alpha\%$ of column A (randomly or based on column B) and sorted its values in ascending order. For the imputation error, we selected $\alpha\%$ of column A (randomly or based on column B) and replaced them with the mean value of column A. $\alpha\%$ is called *error rate*. Note that sorting and imputation errors may either make two columns A and B more independent (random values) or less independent (values in one column selected based on the other). We used random selection for dependence SCs, and column B as a basis for independence SCs. We also explored the combined impact of the two error types. Our *combination error* consists of 80% sorting error and 20% imputation error.

Error-Detection Approaches. We compared SCODED with three state-of-the-art error detection approaches.

`Denial Constraints (DCDetect)` is a DC based error detection approach. The original DC approach marks all the records involved in a DC violation as dirty records [13, 14]. To compare with our top-k error detection algorithms, we extended it as follows. For each record r, we count the number of other records that are inconsistent with r given the DC, and return the top-k records that involve the most number of violations. Table 3 summarizes the DCs used by DCDetect. '×' means that we cannot find a DC to represent the independence SC. As shown in Section 2.2, independence SCs are closely related to EMVDs, and EMVDs are distinct

from Denial Constraints [6, Sec.5.4]. Therefore, *we included DCDetect only in comparison with dependence SCs.*

DCDetect+HC improves the above DCDetect approach for multiple DCs. DCDetect+HC first uses DCDetect to identify candidate dirty records, and then marks the records that are repaired by HoloClean [51] as dirty. This approach has also been adopted in previous work [27].

DBoost [46] is a state-of-the-art outlier detection approach. It was also used by [5] to compare different types of error-detection approaches. We used an implementation available online[9]. We applied DBoost with three models: GMM, Gaussian and Histogram. For categorical data, we employed the bin width that achieves the best f-score results. For numeric data, we employed Gaussian and GMM with the mixture parameter n_subpops threshold set at 3, 0.001,

Approximate Functional Dependency (AFD) [44] leverages approximate integrity constraints. We utilized two AFDs on the HOSP dataset: Zipcode -> City with approximation ratio 25%, and Zipcode -> State with approximation ratio 25%. To use AFDs for top-k error detection, we compare two methods. 1) Following [44], rank each record by projecting its approximation ratio benefit (essentially, counting the number of FD violations due to each record). 2) Our new approach: Translate the AFD into a DSC, as shown in Proposition 2, then apply top-k drill-down to the DSC.

Table 3 summarizes the SCs used by SCODED.Hypothesis tests used the G test for categorical data, and the τ test for numerical data. We implemented the error-drill-down framework, and adopted the K strategy for dependence SCs and the K^c strategy for independence SCs.

Quality Measurement. We considered two user scenarios. (i) The user wants to manually examine a small number of records (e.g., $k = 50$) in order to reason about data errors. For this scenario, since k is fixed, we need to maximize *Precision@K*, which is defined as the ratio of the number of correctly detected records to the number k. (ii) The user wants to detect all errors in order to repair them. For this scenario, as k increases, recall increases while precision potentially decreases. We report *Precision@K*, *Recall@K*, and *F-score@K* by varying k. *Precision@K* is the same as above. *Recall@K* is the ratio of the number of correctly detected records among the returned K records to the number of total erroneous records, and *F-score@K* is their harmonic mean.

To compare with other methods, we apply top-k drill-down analysis regardless of whether the data show a statistically significant violation of an SC. This makes achieving high scores more difficult for SCODED, because it forces it to identify a fixed number of records likely to violate the SC even if the data indicate that the SC is violated only to a small degree.

[9]https://github.com/cpitclaudel/dBoost

Top 1-17

Draft Year	GP>0	Plus-Minus
1998	Yes	0
1999	Yes	0
1999	Yes	0
2000	Yes	0
1999	Yes	0
2004	No	-4
1998	Yes	0
1999	Yes	0
2000	Yes	0
1999	Yes	0
1999	Yes	0
2000	Yes	0
1999	Yes	0
1999	Yes	0
1999	Yes	0
1998	Yes	0
1998	Yes	0

Top 18-34

Draft Year	GP>0	Plus-Minus
1998	Yes	0
1998	Yes	0
1998	Yes	0
1998	Yes	0
2000	Yes	0
2004	No	-4
1999	Yes	0
2004	Yes	-4
2000	Yes	0
2000	Yes	0
2000	Yes	0
1999	Yes	0
1999	Yes	0
1999	Yes	0
1999	Yes	0
1999	Yes	0
2000	Yes	0

Top 35-50

Draft Year	GP>0	Plus-Minus
1999	Yes	0
1998	Yes	0
1999	Yes	0
1999	Yes	0
1999	Yes	0
1999	Yes	0
1999	Yes	0
2000	Yes	0
2007	No	11
1998	Yes	0
1998	Yes	0
2004	Yes	-4
2000	Yes	0
2000	Yes	0
1999	Yes	0
1999	Yes	0

Figure 7: Top-50 results identified by SCODED.

6.2 Case Study

We present two case studies of enforcing SCs on real data.

Model Construction (Hockey). A data scientist wants to build a regression model to predict *Games (GP)*, the total number of games that a player will play after joining NHL. She collects *Hockey* as training data, and starts doing exploratory data analysis for data validation.

Through a Bayesian Network (similar to the example in Figure 1 (b)), she discovers a counter-intuitive SC: given *Draft Year*, the prediction column *Games* strongly depends on *Goal Plus-Minus (GPM)*. Here, *Draft Year* is the year when a player joins a NHL club through the Entry Draft; *GPM* is the plus-minus statistic[10] of a player *before* joining NHL. This strong dependence contradicts previous studies [41].

The data scientist applies SCODED to identify the top-50 records (see Figure 7). She has two surprising observations. First, there are 45 (out of 50) returned records whose *GPM* is equal to 0 while all their *Games* are larger than 0. Second, all those 45 records are from the *Draft Year* before 2000. This points to imputation as a possible cause of the data errors. For early draft years (before 2000), *GPM* has many missing values, and the data provider entered a 0 value even though player's *Games* value is larger than 0. Without detecting this error, learning would infer this (wrong strong) dependence from the training data, leading to low test performance.

Model Testing (Nebraska). A data scientist has constructed a classification model based on historical Nebraska data (1949 - 1969) for *Weather Situation(Weather)* prediction (e.g., rain, snow, fog, etc), and wants to apply it to new data (1970 - 1999) to test model performance. The accepted model shows that there is a strong dependency between the feature *Wind Level (Wind)*, and the label *Weather*. She enforces an approximate SC_1: $\langle \phi_{Wind \not\perp Weather|Year}, \alpha = 0.3 \rangle$ for the test data, so that $p > 0.3$ violates the dependence constraint. Figure 8(a) shows the p-value computation result w.r.t. SC_1 by year.

[10]The difference between the number of goals that a player's team scores versus their opponent's goals when the player is on the ice.

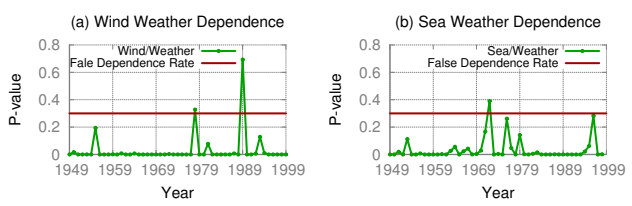


Figure 8: SCs violation detection on Nebraska.

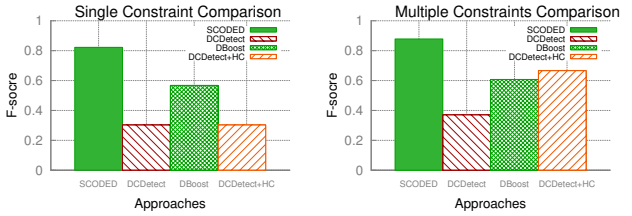


Figure 9: Comparison with DCDetect, DCDetect+HC, and DBoost on Sensor.

SCODED detects two SC violations (1978 and 1989). Drill-down analysis explains why they happend: In all the returned top-50 records for the year 1989 we have *Wind* = 6.07, which turns out to be the imputed value. Referencing the initial data file, it is found that all the data from March to December are missing. The missing value imputation weakens the dependency relationship since knowing *Wind* = 6.07 gives little information about *Weather* on the test data.

Figure 8(b) shows that SC violation also occurred w.r.t. a dependence between *Sea-level Pressure* = Sea and *Weather*. Interestingly, for this feature, the violation was not caused by missing values but by outliers. We find that in the year of 1972 (cf. Figure 8(b)), there are many outliers in January, April, and October. About 64% of those outliers were returned by SCODED. They weaken the dependency relationship since knowing an outlier value of *Sea* gives little information about *Weather* on the test data.

6.3 Evaluation of SCODED Performance

We discuss in detail the performance of SCODED compared to our baselines.

Comparison With DBoost, DCDetect, and DCDetect+HC. We compared SCODED with existing error detection approaches under single and multiple constraints on the *Sensor*.

Single Constraint. Neighboring sensors tend to report similar temperatures, which means that their readings of the temperature should be dependent. We first specified a single SC for Sensor 8 and Sensor 9: $T_8 \not\perp\!\!\!\perp T_9$. For DCDetect+HC and DCDetect, we constructed a corresponding IC as shown in Table 3. Figure 9(a) compares the F-Score of different approaches. We make three observations. i) SCODED achieved a much higher F-Score than DCDetect and DCDetect+HC. This is because that the specified IC did not always hold, which led to many false positives. ii) DCDetect+HC and DCDetect achieved the same F-Score since there is a single constraint. iii) SCODED outperformed DBoost for two reasons. First, DBoost derived correlations from dirty data, and then leveraged the derived correlations to detect errors. However, since data is dirty, the derived correlations might be wrong. Second, DBoost is designed to detect outliers but the dataset has erroneous values (imputed means) that look like a normal value.

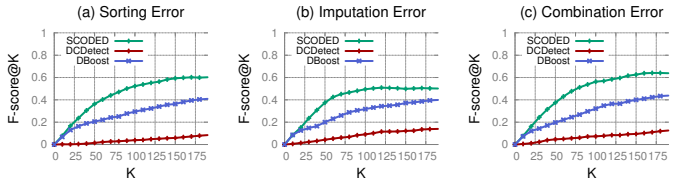


Figure 10: Effectiveness of error detection methods for a dependence SC by varying k (Boston dataset)

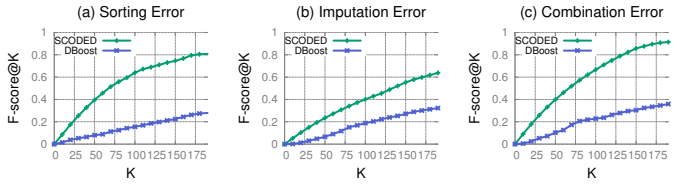


Figure 11: Effectiveness of error detection methods for an independence SC by varying k (Boston dataset)

Multiple Constraints. We specified three SCs between the readings of Sensor 7, Sensor 8, and Senor 9: $T_7 \not\perp\!\!\!\perp T_8$, $T_8 \not\perp\!\!\!\perp T_9$, and $T_7 \not\perp\!\!\!\perp T_9$. For DCDetect+HC and DCDetect, we constructed three corresponding ICs. Figure 9(b) shows the result. We make three observations. i) All four approaches achieved a higher F-score when more constraints (i.e., more user inputs) were provided. ii) SCODED still achieved the highest F-score, which validates the effectiveness of SCs under multiple constraints. iii) DCDetect+HC achieved a higher F-Score than DCDetect since DCDetect+HC leveraged the HoloClean algorithm to holistically infer which records are dirty based on multiple constraints.

Different Forms of SCs. On the *Boston*, for dependence SCs, we compared with DCDetect and DBoost; for independence SCs, since DCDetect cannot express independence relationships (cf. Section 2.2), we compared only with DBoost.

Marginal SCs: $N \not\perp\!\!\!\perp D$ and $R \perp\!\!\!\perp B$. We compared the F-score of SCODED, DCDetect and DBoost using different K values. The average error rate for the N column is moderate (20% – 45%). Results are shown in Figure 10 for $N \not\perp\!\!\!\perp D$ and Figure 11 for $R \perp\!\!\!\perp B$. We can see that SCODED achieved a significantly higher F-score than DCDetect and DBoost for all settings. SCODED's performance depends on the error type: It performed better for sorting error and combination error, where the average F-score is 0.6 and the max F-score is around 0.8. But for the imputation error, the average F-score

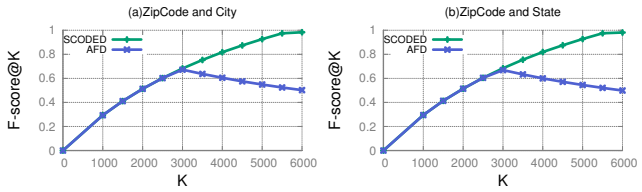


Figure 12: Effectiveness of error detection methods on the Hospital dataset (HOSP dataset).

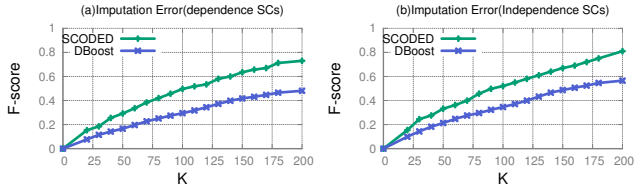


Figure 13: Effectiveness of error detection methods on categorical data (Car dataset)

and the max F-score decrease to 0.5 and 0.6, respectively. This is because the sorting errors have a bigger impact on SCs than imputation errors do.

Conditional SCs: $T \not\perp\!\!\!\perp B \mid C$ *and* $N \perp\!\!\!\perp B \mid T$. We examined the effectiveness of SCODED for *conditional* dependence and independence SCs. The results are similar to unconditional SCs; we omit the details due to the space limit.

Comparison with Functional Dependencies. We compared statistical constraints (SCODED) with functional dependencies (AFD) on the *Hospital* dataset. Figure 12(a) shows the result for Zipcode -> City vs. Zipcode $\not\perp\!\!\!\perp$ City. Figure 12(b) shows the result for Zipcode -> State vs. Zipcode $\not\perp\!\!\!\perp$ State. For this task, DCDetect is equivalent to using AFD [44]. DBoost was not designed for this type of error and produced poor results (not shown).

We make two interesting observations. i) SCODED and AFD attained the same F-score for $K \leq 3000$. This is because both of them achieved 100% precision and the same recall when $K \leq 3000$. ii) SCODED's F-score continued to grow when $K > 3000$ but AFD's F-score started to decrease. This is because AFD did not effectively detect errors on the left-hand side of an AFD, while many errors on the dataset occurred on the left-hand side (e.g., Zipcode).

Effectiveness of Error Detection Approaches on Categorical Data. To evaluate the effectiveness of SCODED on categorical data, we apply the G-test with two SCs ($BP \not\perp\!\!\!\perp Cl$ and $SA \perp\!\!\!\perp DR$) on the *CAR* dataset. DCDetect is not applicable because there are too many violations for the feasible DCs that we constructed. We compared the performance of SCODED and DBoost at the moderate error level. Figure 13 shows the results. Due to the space limit, we limit our focus to imputation errors in this experiment. The average F-score of SCODED and DBoost are 0.49 and 0.25, respectively. The conclusion is that SCODED outperformed DBoost in terms of F-score for both independence and dependence SCs.

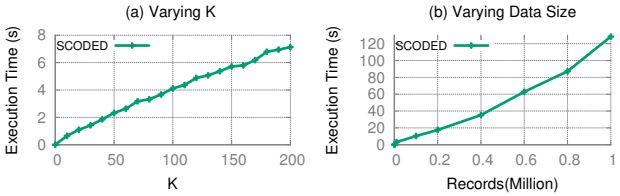


Figure 14: Scalability of SCODED (Boston dataset)

Scalability. We concatenated copies of the Boston dataset to enlarge its data size, and chose the dependence: SC $N \not\perp\!\!\!\perp D$. We examined the execution time of SCODED by varying k and n (# Records), respectively. Recall that the time complexity of SCODED (the k strategy) is $O(n \log n)$ for initialization, and is $O(kn \log n)$ for selecting k records. The results shown in Figure 14 are consistent with this analysis, and demonstrate the good scalability of SCODED in both k and n.

7 RELATED WORK

We review the most closely related work in this section.

Error Detection Methods. User input is invaluable for solving error detection, because user input represents specific domain knowledge. It is therefore important to develop error detection methods for leveraging as many types of user input as possible. Previous error detection methods have been developed to leverage several types of domain knowledge, including value-patterns (e.g., OpenRefine [2] and Trifacta [3]), external knowledge bases or web tables (e.g., Katara [15], Auto-Detect [30], and Uni-Detect [62]), and domain rules, often represented as integrity constraints (e.g., Functional Dependencies [44] and Denial Constraints [13]). A recent trend is to seek user input in the form of binary "clean/dirty" labels. Based on user labelling, error detection can be approached as a classification problem for machine learning (e.g., HoloDetect [27] and Raha [43]). Within these different frameworks for leveraging user input, our paper belongs to the rule-based family. It develops a new kind of rule-based approach, which supports represents user domain knowledge in the form of statistical constraint (SCs).

Outlier detection. Outlier detection typically examines the data distribution of a single column to detect errors (e.g., any datapoint that is more than 3 standard deviation is an outlier) [28]. In contrast, SCs are *multi-column* constraints. Previous work on multi-column outliers includes [18, 19, 46, 52]. The fundamental difference to our work is that outlier detection is driven by the data not by the user, and does not allow a user to specify and analyze a set of SCs.

Previous Research on Statistical Constraints. Previous work shows that SCs are useful and natural for many applications besides error detection. SCs have been studied in

the statistics and AI literature [20, 49]. Existing studies assume that data is clean and explore how to infer SCs from the data. The derived SCs can be used for statistical modeling and causal inference. Ilyas et al. show that SCs (involving pairs of columns) are effective in improving query optimization [32]. Salimi et al. leverage (conditional) independence relationships among attributes to resolve bias in OLAP queries [57]. [62] also employed hypothesis testing on a large corpus of tables to discover diverse errors. Unlike these works, SCODED allows the user to specify SCs explicitly and then detect and explain SC violations. Salimi et al. showed that fairness constraints can be expressed as ISCs [58, 59]. Enforcing fairness can be cast as an ISC detection and repair problem. Their methods are restricted to saturated ISCs and their target application is not error detection.

Data Repairing. Data cleaning involves two important tasks: error detection and data repairing. Our paper focuses on error detection. Data repairing studies the problem of correcting erroneous values. We classify existing works about data repairing into two categories [12].

Data Repair + Error Detection. Given a set of input constraints, they study how to find the minimum change to the data to satisfy the input constraints. Most works assume that the input constraints are ICs [10, 14, 37].

Data Repair After Error Detection. Those approaches uses error detection as a black box [47, 51, 68]. ERACER [47] takes as input a dataset with missing attribute values, and utilizes belief propagation and relational dependency networks to infer the missing values. SCAREd [68] takes as input a dataset with a subset of rows identified as dirty and leverages maximal likelihood for data repairing. HoloClean [51] uses a similar approach as HoloDetect to data repair.

Incremental IC Discovery aims to discover and maintain ICs on dynamic data. Several incremental algorithms, such as DynFD [60] and Swan [7], have been proposed. One of the use cases for SCODED is to enforce SCs on new data. This can be thought of as maintaining SCs on dynamic data. A batch mode solution is to rerun SC violation detection algorithms from scratch whenever data is updated. In future work we will explore incremental on-line versions of SCODED.

Downstream Analysis and Error Explanation. There are some studies on how to clean data for downstream analysis [9, 38, 39] and how to explain data errors [54, 63, 64, 66]. None of them study how to leverage SC for error detection.

8 CONCLUSION AND FUTURE WORK

An SC represents a probabilistic association, or its absence, among columns in a data table. SCs provide a powerful expressive formalism for capturing a user's domain knowledge about statistical and causal relationships. This paper explored how to exploit SCs in data cleaning, by detecting their violations and identifying the underlying data errors. SCs provide an attractive complement to deterministic ICs in mainly two situations: (i) when the user wishes to assert the *ir*relevance of one set of columns to another, and (ii) when the user expects an inferential relationship between columns to hold only approximately to a certain degree, with exceptions. For example, we showed that a functional dependence corresponds to a maximal degree of probabilistic dependence, so weaker dependencies correspond to approximate functional dependencies.

Our SCODED system addresses the challenges of detecting and explaining violations of SCs. For SC violation detection, we showed how statistical metrics can be used to quantify the degree to which an SC is violated. To explain violations, we proposed an error-drill-down framework, and devised efficient algorithms to identify the top-k records that contribute the most to the violation of an SC. We conducted extensive experiments on real-world datasets with both synthetic and real-life errors, as well as a range of constraint types. SCs were shown to be effective in detecting data errors that violate them, compared to state-of-the-art approaches.

Limitations and Future Work. SCs are inherently multi-column constraints and insensitive to single-column errors. Examples include domain constraints (e.g. $age = 200$) and arguably record duplication errors (e.g. Adriana Grande = Ariana Grande). Two important extensions for future work are the following. (1) *Human-in-the-Loop.* Integrate the discovery and validation of SCs to help the user discover and validate them efficiently. (2) *Data Repairing.* Extend SCODED to the error-repairing stage, to automatically repair errors so that the cleaned data satisfies a set of given SCs. For example, we can adapt our methods to search for the top-k cell value corrections that would contribute the most to satisfying a SC. Our current approach is limited to labelling only entire tuples as (likely) dirty, rather than specific values.

SCODED represents a novel approach to leverage powerful statistical methods for error detection. It has great potential for practice, and opens a new set of research directions in the intersection of statistics and data management.

Acknowledgements. We thank reviewers for their useful feedback. This work was supported by Mitacs through an Accelerate Grant, and by NSERC through two discovery grants and a CRD grant, and by Council of Hong Kong through RGC Projects HKU 17229116, 106150091, and 17205115, and by the University of Hong Kong through Projects 104004572, 102009508, and 104004129, and by Innovation and Technology Commission of Hong Kong through ITF project (MRP/029/18). All opinions, findings, and conclusions in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

REFERENCES

[1] 2008. Pandas.DataFrame.Corr API. https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.corr.html. (2008).

[2] 2010. OpenRefine. (2010). http://openrefine.org

[3] 2012. Trifacta. (2012). https://www.trifacta.com

[4] 2018. Working With Data and Machine Learning in Advertising. https://soundcloud.com/talkingmachines/episode-thirteen-working-with-data-and-machine-learning-in-advertising. (2018).

[5] Ziawasch Abedjan, Xu Chu, Dong Deng, Raul Castro Fernandez, Ihab F Ilyas, Mourad Ouzzani, Paolo Papotti, Michael Stonebraker, and Nan Tang. 2016. Detecting Data Errors: Where are we and what needs to be done? *Proceedings of the VLDB Endowment* 9, 12 (2016), 993–1004.

[6] Ziawasch Abedjan, Lukasz Golab, Felix Naumann, and Thorsten Papenbrock. 2018. *Data Profiling*. Vol. 10. Morgan & Claypool Publishers. 1–154 pages.

[7] Ziawasch Abedjan, Jorge-Arnulfo Quiané-Ruiz, and Felix Naumann. 2014. Detecting unique column combinations on dynamic data. In *2014 IEEE 30th International Conference on Data Engineering*. IEEE, 1036–1047.

[8] Paulo RL Almeida, Luiz S Oliveira, Alceu S Britto Jr, and Robert Sabourin. 2018. Adapting dynamic classifier selection for concept drift. *Expert Systems with Applications* 104 (2018), 67–85.

[9] Moria Bergman, Tova Milo, Slava Novgorodov, and Wang Chiew Tan. 2015. Query-Oriented Data Cleaning with Oracles. In *ACM SIGMOD*. 1199–1214.

[10] Philip Bohannon, Michael Flaster, Wenfei Fan, and Rajeev Rastogi. 2005. A Cost-Based Model and Effective Heuristic for Repairing Constraints by Value Modification. In *SIGMOD*. 143–154.

[11] David Maxwell Chickering and Christopher Meek. 2002. Finding Optimal Bayesian Networks.. In *UAI*. 94–102.

[12] Xu Chu, Ihab F Ilyas, Sanjay Krishnan, and Jiannan Wang. 2016. Data cleaning: Overview and emerging challenges. In *Proceedings of the 2016 International Conference on Management of Data*. ACM, 2201–2206.

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

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

[15] Xu Chu, John Morcos, Ihab F Ilyas, Mourad Ouzzani, Paolo Papotti, Nan Tang, and Yin Ye. 2015. Katara: A data cleaning system powered by knowledge bases and crowdsourcing. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. ACM, 1247–1261.

[16] Robert G Cowell, Philip Dawid, Steffen L Lauritzen, and David J Spiegelhalter. 2006. *Probabilistic networks and expert systems: Exact computational methods for Bayesian networks*. Springer Science & Business Media.

[17] Christophe Croux and Catherine Dehon. 2010. Influence functions of the Spearman and Kendall correlation measures. *Statistical methods & applications* 19, 4 (2010), 497–515.

[18] Kaustav Das, Jeff Schneider, and Daniel B Neill. 2008. Anomaly pattern detection in categorical datasets. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 169–176.

[19] Kaustav Das and Jeff G. Schneider. 2007. Detecting anomalous records in categorical datasets. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Jose, California, USA, August 12-15, 2007*. 220–229. https://doi.org/10.1145/1281192.1281219

[20] A Philip Dawid. 1979. Conditional independence in statistical theory. *Journal of the Royal Statistical Society. Series B (Methodological)* (1979), 1–31.

[21] Ryan Elwell and Robi Polikar. 2011. Incremental learning of concept drift in nonstationary environments. *IEEE Transactions on Neural Networks* 22, 10 (2011), 1517–1531.

[22] Ronald Fagin. 1977. Multivalued dependencies and a new normal form for relational databases. *ACM Transactions on Database Systems (TODS)* 2, 3 (1977), 262–278.

[23] Gregory A Fredricks and Roger B Nelsen. 2007. On the relationship between Spearman's rho and Kendall's tau for pairs of continuous random variables. *Journal of statistical planning and inference* 137, 7 (2007), 2143–2150.

[24] Dan Geiger and Judea Pearl. 1993. Logical and algorithmic properties of conditional independence and graphical models. *The Annals of Statistics* (1993), 2001–2021.

[25] Dan Geiger, Thomas Verma, and Judea Pearl. 1990. d-separation: From theorems to algorithms. In *Machine Intelligence and Pattern Recognition*. Vol. 10. Elsevier, 139–148.

[26] David Harrison and Daniel L Rubinfeld. 1978. Hedonic housing prices and the demand for clean air. *Journal of environmental economics and management* 5, 1 (1978), 81–102.

[27] 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*. 829–846.

[28] Joseph M Hellerstein. 2008. Quantitative data cleaning for large databases. *United Nations Economic Commission for Europe (UNECE)* (2008).

[29] David C Howell. 2009. *Statistical methods for psychology*. Cengage Learning.

[30] Zhipeng Huang and Yeye He. 2018. Auto-Detect: Data-Driven Error Detection in Tables. In *Proceedings of the 2018 ACM SIGMOD International Conference on Management of data*. ACM.

[31] Ihab F. Ilyas and Xu Chu. 2015. Trends in Cleaning Relational Data: Consistency and Deduplication. *Foundations and Trends in Databases* 5, 4 (2015), 281–393. https://doi.org/10.1561/1900000045

[32] 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*. ACM, 647–658.

[33] Shawn R Jeffery, Gustavo Alonso, Michael J Franklin, Wei Hong, and Jennifer Widom. 2006. Declarative support for sensor data cleaning. In *International Conference on Pervasive Computing*. Springer, 83–100.

[34] Batya Kenig and Dan Suciu. 2019. Integrity Constraints Revisited: From Exact to Approximate Implication. *arXiv preprint arXiv:1812.09987* (2019).

[35] Nodira Khoussainova, Magdalena Balazinska, and Dan Suciu. 2006. Towards correcting input data errors probabilistically using integrity constraints. In *Proceedings of the 5th ACM international workshop on Data engineering for wireless and mobile access*. ACM, 43–50.

[36] William R Knight. 1966. A computer method for calculating Kendall's tau with ungrouped data. *J. Amer. Statist. Assoc.* 61, 314 (1966), 436–439.

[37] Solmaz Kolahi and Laks V. S. Lakshmanan. 2009. On approximating optimum repairs for functional dependency violations. In *Database Theory - ICDT 2009, 12th International Conference, St. Petersburg, Russia, March 23-25, 2009, Proceedings*. 53–62.

[38] Sanjay Krishnan, Michael J Franklin, Ken Goldberg, Jiannan Wang, and Eugene Wu. 2016. Activeclean: An interactive data cleaning framework for modern machine learning. In *Proceedings of the 2016 International Conference on Management of Data*. ACM, 2117–2120.

[39] Sanjay Krishnan, Jiannan Wang, Michael J Franklin, Ken Goldberg, Tim Kraska, Tova Milo, and Eugene Wu. 2015. SampleClean: Fast and Reliable Analytics on Dirty Data. *IEEE Data Eng. Bull.* 38, 3 (2015), 59–75.

[40] Yejia Liu, Oliver Schulte, and Chao Li. 2018. Model Trees for Identifying Exceptional Players in the NHL and NBA Drafts. In *International Workshop on Machine Learning and Data Mining for Sports Analytics*. Springer, 93–105.

[41] Brian Macdonald. 2011. A Regression-Based Adjusted Plus-Minus Statistic for NHL Players. *Journal of Quantitative Analysis in Sports* 7, 3 (2011), 29.

[42] Samuel R Madden, Michael J Franklin, Joseph M Hellerstein, and Wei Hong. 2005. TinyDB: an acquisitional query processing system for sensor networks. *ACM Transactions on database systems (TODS)* 30, 1 (2005), 122–173.

[43] Mohammad Mahdavi, Ziawasch Abedjan, Raul Castro Fernandez, Samuel Madden, Mourad Ouzzani, Michael Stonebraker, and Nan Tang. 2019. Raha: A Configuration-Free Error Detection System. In *Proceedings of the 2019 International Conference on Management of Data*. ACM, 865–882.

[44] 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, 355–363.

[45] Dimitris Margaritis. 2003. *Learning Bayesian network model structure from data*. Technical Report. Carnegie-Mellon Univ Pittsburgh Pa School of Computer Science.

[46] Zelda Mariet, Rachael Harding, Sam Madden, et al. 2016. Outlier Detection in Heterogeneous Datasets using Automatic Tuple Expansion. (2016).

[47] Chris Mayfield, Jennifer Neville, and Sunil Prabhakar. 2010. ERACER: a database approach for statistical inference and data cleaning. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*. ACM, 75–86.

[48] Mathias Niepert, Marc Gyssens, Bassem Sayrafi, and Dirk Van Gucht. 2013. On the conditional independence implication problem: A lattice-theoretic approach. *Artificial Intelligence* 202 (2013), 29–51.

[49] J. Pearl. 2000. *Causality: Models, Reasoning, and Inference*. Cambridge university press.

[50] Judea Pearl. 2014. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Elsevier.

[51] Theodoros Rekatsinas, Xu Chu, Ihab F. Ilyas, and Christopher Ré. 2017. HoloClean: Holistic Data Repairs with Probabilistic Inference. *PVLDB* 10, 11 (2017), 1190–1201. http://www.vldb.org/pvldb/vol10/p1190-rekatsinas.pdf

[52] Fatemeh Riahi and Oliver Schulte. 2015. Model-based outlier detection for object-relational data. In *Computational Intelligence, 2015 IEEE Symposium Series on*. IEEE, 1590–1598.

[53] Saharon Rosset, Claudia Perlich, Grzergorz Świrszcz, Prem Melville, and Yan Liu. 2010. Medical data mining: insights from winning two competitions. *Data Mining and Knowledge Discovery* 20, 3 (2010), 439–468.

[54] Sudeepa Roy and Dan Suciu. 2014. A formal approach to finding explanations for database queries. In *SIGMOD*. 1579–1590.

[55] Cory J. Butz S. K. Michael Wong and Dan Wu. 2000. On the implication problem for probabilistic conditional independency. In *IEEE Trans. Systems, Man, and Cybernetics, Part A*. 30(6):785–805.

[56] Babak Salimi, Corey Cole, Peter Li, Johannes Gehrke, and Dan Suciu. 2018. HypDB: a demonstration of detecting, explaining and resolving bias in OLAP queries. *Proceedings of the VLDB Endowment* 11, 12 (2018), 2062–2065.

[57] Babak Salimi, Johannes Gehrke, and Dan Suciu. 2018. Bias in OLAP Queries: Detection, Explanation, and Removal. In *ACM SIGMOD*. 1021–1035.

[58] Babak Salimi, Luke Rodriguez, Bill Howe, and Dan Suciu. 2019. Capuchin: Causal database repair for algorithmic fairness. *arXiv preprint arXiv:1902.08283* (2019).

[59] Babak Salimi, Luke Rodriguez, Bill Howe, and Dan Suciu. 2019. Interventional Fairness: Causal Database Repair for Algorithmic Fairness. In *Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019*. 793–810. https://doi.org/10.1145/3299869.3319901

[60] 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*. 253–264.

[61] Milan Studeny. 1990. Conditional independence relations have no finite complete characterization. (1990).

[62] Pei Wang and Yeye He. 2019. Uni-Detect: A Unified Approach to Automated Error Detection in Tables. In *Proceedings of the 2019 International Conference on Management of Data*. ACM, 811–828.

[63] Xiaolan Wang, Xin Luna Dong, and Alexandra Meliou. 2015. Data X-Ray: A Diagnostic Tool for Data Errors. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015*. 1231–1245. https://doi.org/10.1145/2723372.2750549

[64] Xiaolan Wang, Alexandra Meliou, and Eugene Wu. 2017. QFix: Diagnosing errors through query histories. In *Proceedings of the 2017 ACM International Conference on Management of Data*. ACM, 1369–1384.

[65] Larry Wasserman. 2013. *All of statistics: a concise course in statistical inference*. Springer Science & Business Media.

[66] Eugene Wu and Samuel Madden. 2013. Scorpion: Explaining away outliers in aggregate queries. *Proceedings of the VLDB Endowment* 6, 8 (2013), 553–564.

[67] Weichao Xu, Yunhe Hou, YS Hung, and Yuexian Zou. 2013. A comparative analysis of Spearman's rho and Kendall's tau in normal and contaminated normal models. *Signal Processing* 93, 1 (2013), 261–276.

[68] Mohamed Yakout, Laure Berti-Équille, and Ahmed K Elmagarmid. 2013. Don't be SCAREd: use SCalable Automatic REpairing with maximal likelihood and bounded changes. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. ACM, 553–564.

[69] Jing Nathan Yan, Oliver Schulte, MoHan Zhang, Jiannan Wang, and Reynold Cheng. 2020. SCODED: Statistical Constraint Oriented Data Error Detection. http://tiny.cc/sigmod2020-scoded. *Technical Report* (2020).