Going Beyond Provenance: Explaining Query Answers with Pattern-based Counterbalances

Zhengjie Miao[*]
Duke University
zjmiao@cs.duke.edu

Qitian Zeng[*]
IIT
qzeng3@hawk.iit.edu

Boris Glavic
IIT
bglavic@iit.edu

Sudeepa Roy
Duke University
sudeepa@cs.duke.edu

ABSTRACT

Provenance and intervention-based techniques have been used to explain surprisingly high or low outcomes of aggregation queries. However, such techniques may miss interesting explanations emerging from data that is *not* in the provenance. For instance, an unusually low number of publications of a prolific researcher in a certain venue and year can be explained by an increased number of publications in another venue in the same year. We present a novel approach for explaining outliers in aggregation queries through *counterbalancing*. That is, explanations are outliers in the opposite direction of the outlier of interest. Outliers are defined w.r.t. patterns that hold over the data in aggregate. We present efficient methods for mining such *aggregate regression patterns* (*ARPs*), discuss how to use ARPs to generate and rank explanations, and experimentally demonstrate the efficiency and effectiveness of our approach.

ACM Reference Format:
Zhengjie Miao, Qitian Zeng, Boris Glavic, and Sudeepa Roy. 2019. Going Beyond Provenance: Explaining Query Answers with Pattern-based Counterbalances. In *2019 International Conference on Management of Data (SIGMOD '19), June 30-July 5, 2019, Amsterdam, Netherlands.* ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3299869.3300066

1 INTRODUCTION

In today's data-driven world, modern data analysis systems provide a multitude of advanced tools for prediction and association, multi-dimensional data aggregation, and sophisticated visualization support. Typically, when users are trying to understand trends or patterns by aggregating data, outliers may be revealed, or, the user may simply wonder why an observed value is higher or lower than her expectation. The most obvious approach to explain query results of interest is to output the tuples in the result's *provenance*, i.e., the tuples that contributed to the generation of the query answer. Provenance has been studied in database research traditionally for non-aggregate queries [8, 10, 12, 17], but also for aggregate queries [6, 7]. However, often data provenance is not sufficient to explain the existence of outliers in query answers as illustrated in the following example.

Pub

author	pubid	year	venue
A_X	P1	2005	SIGKDD
A_X	P2	2004	SIGKDD
A_Y	P2	2004	SIGKDD
A_Z	P3	2004	SIGMOD

Table 1: Running example publications dataset

Example 1. *We use a simplified version of the DBLP (http://dblp.uni-trier.de/) publication dataset with schema* `Pub(author, pubid, year, venue)` *as our running example. An (anonymized) sample is shown in Table 1 (the full DBLP dataset consists of several million records). The following SQL query Q_0 computes the number of publications per author, year, and publication venue.*

```
SELECT author , year , venue , count(*) AS pubcnt
FROM Pub
GROUP BY author , year , venue
```

Table 2 shows a subset of this query's result for author A_X. Given such a result, a user may be interested in knowing why certain rows have a higher/lower aggregation function value than expected. For instance, a user who is aware that author A_X is a prolific data mining researcher may wonder "why did A_X publish only 1 paper in SIGKDD in 2007?".

The provenance of the query result t_0 = (A_X, SIGKDD, 2007, 1) in the above example would enumerate the input tuples based on which the result was computed – in this case, the one SIGKDD paper A_X published in 2007. However, additional details about this paper do not help us to understand why the number of A_X's publications in SIGKDD in this year is low.

The problem of explaining such "why high/low" questions for aggregate query answers has been recently studied [35, 36, 47]. These approaches apply explanation by 'intervention' – if by removing a subset of the input tuples the answer

[*]denotes equal contribution

ACM ISBN 978-1-4503-5643-5/19/06…$15.00
https://doi.org/10.1145/3299869.3300066

author	venue	year	pubcnt
A_X	SIGKDD	2006	4
A_X	SIGKDD	2007	1
A_X	SIGKDD	2008	4
A_X	VLDB	2006	4
A_X	VLDB	2007	4
A_X	VLDB	2008	1
A_X	ICDE	2006	6
A_X	ICDE	2007	6
A_X	ICDE	2008	4

Table 2: Partial result of query Q_0 from Example 1

Rank	Explanation	score
1	(A_X, ICDE, 2007, 6)	13.78
2	(A_X, ICDE, 2006, 6)	10.91
3	(A_X, ICDM, 2007, 5)	6.44
4	(A_X, VLDB, 2007, 4)	5.77
5	(A_X, SIGMOD, 2008, 4)	5.57
6	(A_X, TKDE, 2006, 4)	4.95
7	(A_X, ICDM, 2008, 5)	4.70
8	(A_X, VLDB, 2006, 4)	4.57
9	(A_X, ICDE, 2008, 4)	3.97
10	(A_X, 2010, 63)	3.20

Table 3: Top-10 explanations returned by Cape for the question ϕ_0 = *"why is the number of A_X's SIGKDD 2007 publications low?"* based on $Q_0(Pub)$ from Example 1

changes in the opposite direction, then those tuples provide an explanation for the outcome. The goal is then to use predicates (selection) to succinctly describe those tuples. For instance, using the data from Example 1, explanations for A_X's high count of ICDE publications in 2006 are predicates describing tuples whose deletion would reduce the number of A_X's publications in ICDE 2006, e.g., papers published with a specific co-author.

Although the frameworks in [35, 36, 47] provide insightful explanations, they are ineffective for the example shown above, because they only consider tuples appearing in the provenance of the specified query answer(s). *That is, these methods cannot find explanations emerging from data that does not belong to the provenance, which can only be found if we look at the bigger context.*

We propose a new approach for explaining aggregate query answers and present a system called Cape **(Counterbalancing with Aggregate Patterns for Explanations)** that finds explanations beyond provenance with the help of patterns that hold on the data. For instance, in Example 1, by asking the question ϕ_0 = *"why is the number of A_X's SIGKDD 2007 publications only 1"?*, we make an implicit assumption that there is some underlying *pattern*, i.e., *A_X typically publishes roughly 3-5 SIGKDD papers per year*, which holds on the data in general, but is violated by the result of interest. We refer to this type of patterns as **aggregate regression patterns** (**ARPs**), because they describe a trend observed in the result of a group-by aggregation query. We mine such ARPs and use them to provide explanations *counterbalancing* the user's observation. That is, we find data points which are outliers in the opposite direction with respect to a pattern related to the user question.

Table 3 shows the top-10 explanations produced by our framework for question ϕ_0 ranked according to their scores (we will introduce the scoring function in Section 3.3). The top explanations include ICDE 2007 and ICDE 2006 publications by A_X, suggesting that he might have sent more papers to ICDE in these years instead of to SIGKDD in 2007. The ranking of the explanations illustrate their relative importance in explaining the user question. A large number of ICDE publications by A_X is more 'surprising' than ICDM publications since the primary research of A_X is in the area of data mining. The higher numbers of publications in other venues in 2007 are more relevant explanations to the user question involving SIGKDD 2007 compared to publications in the adjacent years 2006 or 2008, which in turn are ranked higher than the publications in years that are farther apart like 2010. The top-10 explanations also include 63 publications by A_X in year 2010, suggesting he had fewer SIGKDD 2007 papers possibly because he published a much higher number of papers than usual in 2010, albeit with a low score since 2010 is not adjacent to 2007. This also illustrates that the explanations returned by Cape not only emanate from the answers of query Q_0, and in general, can have coarser (or finer) schema depending on the originating ARP (see Section 3). Years/venues not appearing in top explanations suggest that they are possibly 'as expected'. *Note that none of the explanations appearing in Table 3 would be returned by the approaches from [35, 36, 47] since they are disjoint from the provenance of SIGKDD publications of A_X in 2007.*

We make the following contributions in this paper.

(1) **ARPs**: We formalize *aggregate regression patterns (ARPs)*, which model trends that hold in the result of aggregation queries using regression techniques [32]. These patterns provide the critical context needed to find explanations beyond tuples in the provenance of answers of interest. An ARP partitions the output of an aggregation using a subset of the group-by attributes, and within each partition models the relationship between the remaining group-by attributes and the aggregation function result using regression (Section 2).

(2) **Counterbalancing explanations using ARPs**: We discuss how explanations *counterbalancing* the observation of interest can be obtained using ARPs. To find explanations that are related to the user question with respect to the patterns, we introduce two operations: *refinement* and *drill-down.* We develop a scoring function for ranking explanations which incorporates outlierness (the deviation from

expected values) and similarity (distance from the observation specified in the user question). Moreover, we discuss how to output top-k explanations efficiently by pruning the search space based on an upper bound for the scoring function (Section 3).

(3) **Mining ARPs**: We present an algorithm for mining ARPs and discuss optimizations that include using functional dependencies to omit spurious patterns and reusing query results during pattern mining (Section 4).

(4) **Experimental evaluation**: We implement our approach as a system called Cape and demonstrate its effectiveness and efficiency through experiments and a user study (Section 5 and Appendix B).

2 AGGREGATE REGRESSION PATTERNS

We now introduce the type of user questions supported by our approach and define aggregate regression patterns. Let R be a relation with schema $R = (A_1, \ldots, A_n)$ over a universal domain of values $\mathcal{U}$. Given a tuple t with attributes T, for a subset of attributes $S \subseteq T$ we use $t[S]$ to denote the projection of t on attributes S. Let Q be an *aggregate query* of the form $\gamma_{\mathtt{G},\mathtt{agg(A)}}(R)$, where G is a set of group by attributes ($G \subseteq R$), $A \in R$, and *agg* is an aggregate function (e.g., `sum`). In SQL such a query can be written as:

```
SELECT G, agg(A) FROM R GROUP BY G
```

Given such a query, a user may consider a particular aggregation result to be unusually low or high and may wonder why this is the case. We call such types of requests *user questions* in this work, which we formally define below.

Definition 1 (user question). *Let R be a relation. A* user question *(UQ) is a quadruple $\phi = (Q, R, t, dir)$ where Q is an aggregate query; $t \in Q(R)$; and $dir \in \{high, low\}$ is a direction specifying whether the aggregation result $t[agg(A)]$ is higher or lower than expected.*

For instance, reconsider Example 1. If the user asks *"Why did A_X publish only 1 paper in SIGKDD in 2007?"* based on the answer of Q_0, then the corresponding user question will be $\phi_0 = (Q_0, Pub, t_0, low)$, where t_0 is the tuple $(A_X, SIGKDD, 2007, 1)$ from Table 2.

The Cape framework we present in this paper explains such user questions using aggregate regression patterns (ARPs). We first briefly review regression models (Section 2.1), then introduce ARPs explaining when a pattern holds locally and globally (Sections 2.2 and 2.3), and finally define the ARP-mining problem.

2.1 Regression Analysis

We use regression analysis to model trends in data. In machine learning, regression is used to estimate the relationship between a dependent variable and one or more independent variables. A regression model estimates the dependent variable as a function of the independent variables. Such a *prediction function* $g : \mathcal{X} \rightarrow \mathcal{Y}$ is *learned* from a training dataset containing samples consisting of a value for the dependent variable (whose domain is $\mathcal{Y}$) and values for all independent variables (whose domain is $\mathcal{X}$). Many regression models have been proposed in the literature [13]. We use *linear regression* (denoted by `Lin`, where g is a linear function) and *constant regression* (denoted by `Const`, where g is a constant function). However, most of our results are independent of what type of regression is used. The choice of linear and constant regression was informed by the fact that these models are easy to explain to users.[1]

The **goodness-of-fit** of a regression model describes how well it fits a dataset, and it is measured based on the discrepancy between observed values and the values predicted under the regression model. Given a training dataset $\mathcal{D} \subseteq \mathcal{X} \times \mathcal{Y}$, a goodness-of-fit measure is a function

$$GoF_{\mathcal{D}} : \mathbb{D}_{Model} \rightarrow [0, 1]$$

that takes a regression model g, and returns a value from $[0, 1]$, where $\mathbb{D}_{Model}$ denotes the set of all possible regression models. We require that $GoF(g) = 1$ if and only if $\forall x \in \mathcal{X} : g(x)$ is equal to the observed value in $\mathcal{D}$. For simplicity, we will use GoF instead of $GoF_{\mathcal{D}}$ when $\mathcal{D}$ is clear from the context. We use standard goodness-of-fit measures: *Pearson's chi-square test* [34] for constant regression, and the *R-squared (R^2) statistic* [13] for linear regression. In contrast to the common use of regression for classification, we use regression to determine whether a trend in the data can be described using a particular type of regression model. Hence, we fit the regression model over the full dataset.

2.2 Local Patterns

We now introduce aggregate regression patterns (or ARPs for short) that model trends in the result of an aggregation query and are used to explain user questions. Consider the relation `Pub(author, pubid, year, venue)`. Informally, aggregate regression patterns capture patterns such as:

P_1 : = *"for each author, the number of publications is constant over the years"*

This pattern states that if we partition the rows in the output of the query $\gamma_{\mathtt{author,year,count(pubid)}}(Pub)$ based on their `author` names, then within each fragment the relationship between the `year` values and the aggregate output `count(pubid)` (number of publications) can be described approximately though a constant function.

[1]While linear regression models with multiple independent variables are hard to visualize, they can still be described quantitatively by explaining which inputs affect the output positively or negatively and to what degree.

As illustrated in the example above, an ARP partitions the query result based on a subset $F \subseteq G$ of the group-by attributes and within each fragment describes correlations between the remaining group-by attributes $V = (G - F)$ and the results of the aggregation function. The **partition attributes** F are the attributes on which we partition the output of the query. In P_1, there is a single partition attribute $F = \{\texttt{author}\}$. The **predictor attributes** V are the attributes that we use as the predictor (or, independent) variables in the regression. In P_1, $V = \{\texttt{year}\}$. In addition to the partition and predictor attributes, an ARP also specifies an aggregate function to be used (`count` in P_1), which attribute we are aggregating over (in P_1 this is not applicable, since `count` is used), and what regression model type to use (`Const`, i.e., constant regression in P_1).

Definition 2 (ARP). *Given a relation R, an* **aggregate regression pattern (ARP)** *is a tuple*

$$P = (F, V, agg, A, M)$$

where (i) $F, V \subset R, F \neq \emptyset, V \neq \emptyset, F \cap V = \emptyset$, (ii) agg is an aggregate function (one of `count`, `sum`, `min`, `max`*); (iii) $A \in R$ and $A \notin F \cup V$ (A is $*$ when agg is* `count`*); (iv) M is a regression model type.*

For convenience we also use an alternative notation for patterns (leaving out M if it is clear from the context or irrelevant to the discussion):

$$[F] : V \overset{M}{\rightsquigarrow} agg(A)$$

Using this notation the example pattern is written as:

$$P_1 = [\texttt{author}] : \texttt{year} \overset{\texttt{Const}}{\rightsquigarrow} \texttt{count}(*)$$

For an ARP $P = [F] : V \rightsquigarrow agg(A)$ and relation R, we use $frag(R, P)$ to denote the fragments of R according to P, i.e., the set of distinct partition attribute values appearing in R:

$$frag(R, P) = \pi_F(R).$$

In the following, we will use the term fragment to either mean a partition attribute value f or the corresponding subset of the query result where $F = f$. The meaning will be clear from the context. Given $f \in frag(R, P)$, we define the **retrieval query** $Q_{P,f}$ which computes the fragment f:

$$Q_{P,f} := \gamma_{\mathsf{F \cup V, agg(A)}}(\sigma_{F=f}(R))$$

The result of the retrieval query $Q_{P,f}$ for $f \in frag(R, P)$ is used to train a regression model for this fragment mapping V to $agg(A)$. Using the instance of table Pub shown in Figure 1, we have $frag(Pub, P_1) = \{A_X, A_Y, A_Z\}$. Retrieval query results are shown on the right in this figure. We use colors to indicate which fragments of the input and output correspond to which partition attribute value.

Given an ARP P, we want to check whether P correctly describes the behavior of the data within a fragment f of the

Pub

	author	pubid	year	venue
	A_X	P1	2004	SIGKDD
	A_X	P2	2004	SIGKDD
A_X	A_X	P3	2005	SIGKDD
	A_X	P4	2005	SIGKDD
	A_X	P5	2005	ICDE
	A_Y	P2	2004	SIGKDD
	A_Y	P6	2004	ICDE
A_Y	A_Y	P7	2004	ICDM
	A_Y	P8	2005	ICDE
A_Z	A_Z	P9	2004	SIGMOD

Q_{P_1,A_X}

year	cnt	g_{P_1,A_X}
2004	2	2.5
2005	3	2.5

Q_{P_1,A_Y}

year	cnt	g_{P_1,A_Y}
2004	3	2
2005	1	2

Q_{P_1,A_Z}

year	cnt	$\lvert Q_{P_1,A_Z}(R)\rvert$
2004	2	$< \delta$

Figure 1: Evaluating whether pattern P_1 holds

query result. If that is the case then we consider the pattern to "*hold locally*" for f. In particular, we deem a pattern to hold locally if (1) there is sufficient evidence in the data, i.e., if there are sufficiently many distinct values for the predictor attributes (above a **local support threshold** δ) in fragment f; and (2) we can fit a regression model with a sufficiently high goodness-of-fit (called **local model quality**) which within the fragment predicts the aggregate value based on the values of V. Assume that we set the local support threshold $\delta = 2$. Then for the example shown in Figure 1 this disqualifies $author = A_Z$, because there is only one distinct value (2004) for the predictor attribute (year).

Definition 3. *Given a relation R, a pattern $P = [F] : V \overset{M}{\rightsquigarrow} agg(A)$* **holds locally** *on $f \in frag(R, P)$ with goodness-of-fit threshold θ and local support threshold δ (written as $f \models_{\theta,\delta} P$) if there exists a regression model g of type M such that*

$$GoF(g) \geq \theta \qquad \textbf{(local model quality)}$$

$$|Q_{P,f}(R)| \geq \delta \qquad \textbf{(local support)}$$

We will use $g_{P,f}$ to denote the regression model based on which P was determined to hold locally on $f \in frag(R, P)$ (denoted by g the definition above). We assume that g is determined though a fitting algorithm that identifies a particular model, ideally one with a maximal achievable GoF value. We now further illustrate this concept by example.

Example 2. *Continuing with the example shown in Figure 1, we obtain a regression model g for each fragment whose prediction function is of the form $g(x) = \beta$ for some constant β (the model chosen in P_1 is* `Const`*). Let us assume that $\theta = 0.2$ and that the constant regression models we have trained over the fragments for A_X and A_Y have goodness-of-fit higher than 0.2 and, thus, that the pattern P_1 holds locally for these fragments. Recall that $g_{P,f}$ denotes the model based on which pattern P holds locally for f. We show the values predicted by these models on the right of the fragments in Figure 1. That is, the models predict that author A_X publishes 2.5 papers each year while author A_Y publishes about 2 papers per year.*

2.3 Global Patterns

As shown in the example above, a pattern may only hold locally for some fragments. For instance, for many authors the number of publications per year increases roughly linearly over the years, but this is certainly not true for all authors, e.g., an author may leave academia. We introduce the notion of a **pattern holding globally** to model the descriptiveness of the pattern for the whole dataset.

To validate the universality of a pattern, we take into account the fraction of fragments from $frag(R, P)$ for which the pattern P holds locally. We refer to this measure as the **global confidence**. We discard patterns for which the global confidence is below a threshold λ. To calculate the global confidence of a pattern we only consider $f \in frag(R, P)$ for which the support is high enough ($|Q_{P,f}(R)| \geq \delta$). The rationale behind excluding fragments with support less than δ from the calculation is that for such a fragment we do not have sufficient information to determine with confidence whether the pattern holds locally. To limit the search space, and to only retrieve patterns that hold for a sufficiently large number of fragments, we consider an additional minimum **global support threshold** Δ; the **global support** is the number of fragments for which the pattern holds locally.

Definition 4. *Given local model quality threshold $\theta \in [0, 1]$, local support threshold $\delta \in \mathbb{N}$, global confidence threshold $\lambda \in [0, 1]$, and global support threshold $\Delta \in \mathbb{N}$, a pattern $P = [F] : V \rightsquigarrow agg(A)$* **holds globally** *over a relation R for local thresholds θ, δ and global thresholds λ, Δ, which we denote by $R \models_{(\theta,\delta),(\lambda,\Delta)} P$, if:*

$$\frac{|frag_{good}|}{|frag_{supp}|} \geq \lambda \quad \textbf{global confidence} \qquad |frag_{good}| \geq \Delta \quad \textbf{global support}$$

where

$$frag_{good} = \{f \mid f \in frag(R, P) \wedge f \models_{\theta,\delta} P\}$$
$$frag_{supp} = \{f \mid f \in frag(R, P) \wedge |Q_{P,f}(R)| \geq \delta\}$$

In the above definition, $frag_{good}$ is the subset of $frag(R, P)$ for which the pattern holds locally and $frag_{supp}$ is the subset of $frag(R, P)$ for which the local support is high enough. Note that instead of $|frag(R, P)|$, the denominator in the definition of global confidence is $|frag_{supp}|$, because we only include fragments for which we have enough evidence so that fitting a regression model is meaningful.

In Example 2, we evaluated whether Pattern P_1 holds locally for all fragments with a local support above the threshold. We decide whether the pattern *"for every author, the number of publications is constant over the years"* holds globally based on its global confidence and global support. For example, if we set $\lambda = 0.5$ and $\Delta = 2$ then the pattern holds, because there are $2 \geq \Delta$ fragments (A_X and A_Y) for which there is sufficient support ($\geq \delta = 2$) and for both of these authors the pattern holds locally, i.e., $\frac{|frag_{good}|}{|frag_{supp}|} = 1 \geq \lambda = 0.5$.

The ARP mining problem: Below we state the problem of detecting all patterns that globally hold over a relation R; these patterns are used as an input for finding explanations using counterbalances as discussed in the next section.

The **ARP mining problem** is defined as follows:

- **Input:** Relation R, local thresholds θ, δ, global thresholds λ, Δ
- **Output:** $\mathcal{P} = \{P \mid R \models_{(\theta,\delta),(\lambda,\Delta)} P\}$

As we discuss in Section 4, this problem is solvable in PTIME in *data complexity* [45] (a fixed schema), whereas the number of patterns can be exponential in the size of the schema (e.g., when the thresholds are trivial such that all possible subsets F, V of R satisfy the constraints). Thus, simply listing the answers may need exponential time in the size of the schema of R. In spite of the polynomial data complexity, a naive algorithm is not efficient for practical purposes. In Section 4 we discuss efficient ARP-mining algorithms.

3 EXPLANATIONS

In response to a user question $\phi = (Q, R, t, dir)$ (Definition 1), our approach returns a list of candidate explanations based on ARPs. In this section, we assume that the parameters (θ, δ), (λ, Δ), and a set $\mathcal{P}$ of ARPs that hold globally on the given relation R w.r.t. these parameters are given as input. As discussed in Section 2, we assume that the regression models based on which the patterns were determined to hold locally are made available. We will focus on explanations by *counterbalance* as introduced in the introduction. Our approach consists of finding *relevant* patterns that generalize the user question in the sense that $F \cup V$ is a subset of the group-by attributes from the question. Starting from such a pattern, we find *refinements* that are patterns specializing a relevant pattern by subdividing its fragments, and use pairs of relevant patterns and their refinements to drill down into parts of fragments that explain the unusual outcome by providing counterbalance. For instance, A_X's publication count in 2007 was low (user question), possibly because his publication counts in 2006 and 2008 were high (counterbalance).

3.1 Relevant Patterns

A pattern P may be **relevant for a user question** $\phi = (Q, R, t, dir)$ where $Q = \gamma_{\mathrm{G},\mathrm{agg(A)}}(R)$ if it describes a trend that holds over the part of relation R (rows and attributes) corresponding to the user question. For example, if the group-by attributes in the user question are $G = \{\texttt{author}, \texttt{year}\}$, then patterns that describe trends over a subset of these attributes (a superset of the data in the provenance of the tuple t in ϕ) may be considered relevant.

Definition 5. *A pattern P with partition attributes F and predictor attributes V is called* ***relevant for a user question***

Input Relation Pub

author	pubid	year	venue
A_X	P1	2006	SIGKDD
A_X	P2	2006	SIGKDD
A_X	P3	2006	SIGKDD
A_X	P4	2006	ICDE
A_X	P5	2007	SIGKDD
A_X	P6	2007	ICDE
A_X	P7	2007	ICDE
…	…	…	…

(2) Relevant Pattern and User Question Fragment

$[\mathsf{author}] : \mathsf{year} \overset{\mathsf{Const}}{\rightsquigarrow} \mathsf{count}(*)$

author	year	cnt
A_X	2006	4
A_X	2007	3
…	…	…

(1) User Query Result and Tuple

author	year	venue	cnt
A_X	2006	SIGKDD	4
A_X	2006	ICDE	1
$t_0 \rightarrow$ A_X	2007	SIGKDD	1
A_X	2007	ICDE	2
…	…	…	…

(3) Refined Pattern and Counterbalance

$[\mathsf{author}, \mathsf{venue}] : \mathsf{year} \overset{\mathsf{Const}}{\rightsquigarrow} \mathsf{count}(*)$

author	year	venue	cnt
A_X	2006	SIGKDD	4
A_X	2006	ICDE	1
A_X	2007	SIGKDD	1
A_X	2007	ICDE	2
…	…	…	…

Figure 2: Finding an explanation. The user question tuple is highlighted in dark grey and the explanation (counterbalance) w.r.t. the refined pattern in light grey

$\phi = (Q, R, t, dir)$ *with group-by attributes* G *if:*

$$(F \cup V) \subseteq G \qquad \textbf{(Generalizes } \phi\textbf{)}$$

$$t[F] \models_{\theta,\delta} P \qquad \textbf{(Holds locally for } \phi\textbf{)}$$

Intuitively, (i) if $F \cup V \subseteq G$, then the pattern describes a trend that holds over some user question attributes, and (ii) if $t[F] \models_{\theta,\delta} P$, then the trend described by the pattern is confirmed by the fragment containing the user question tuple. The following example further illustrates this concept.

Example 3. *Consider a modified version of our running example shown in Figure 2 and the* user question $\phi_0 = (Q_0, Pub, t_0, low)$ *for* $Q_0 = \gamma_{\mathsf{author},\mathsf{venue},\mathsf{year},\mathsf{count}(*)}$ *and* $t_0 = (A_X, \mathsf{SIGKDD}, 2007, 1)$*. To explain why* A_X *published only one paper in SIGKDD in 2007, we consider patterns that describe trends over a subset of the question's group-by attributes. One such pattern is* $P_1 = [\mathsf{author}] : \mathsf{year} \overset{\mathsf{Const}}{\rightsquigarrow} \mathsf{count}(*)$ *from Section 2. To test whether this pattern is a viable candidate for creating explanations for* ϕ_0*, we first test whether* P_1 *holds locally for* $f = t_0[author] = (A_X)$ *using the fragment for* A_X *which is shown on the top right in Figure 2. Assuming that this is the case, we consider* P_1 *relevant for* ϕ_0*.*

3.2 Pattern Refinement and Explanations

A pattern P that is relevant to a user question may be used to explain the question. We already know that P holds locally for the fragment corresponding to the user question. That is, the data corresponding to the question conforms to the trend described by the pattern, e.g., A_X publishes roughly a constant number of papers every year. To explain why the aggregation value of the user question is unusually high or low, we then drill down into P to find related observations from the input data, which according to P or a more specific pattern P' (a *refinement* of P) deviate in the opposite direction (i.e., provide a counterbalance). For instance, if the user question is asking for explanations for a high outcome, then we find low outcomes according to a refined pattern.

Definition 6 (Pattern refinement). *Given a pattern* $P = [F] : V \overset{M}{\rightsquigarrow} agg(A)$ *and user question* $\phi = (Q, R, t, dir)$ *with group-by attributes* G*, we call a pattern* $P' = [F'] : V \overset{M'}{\rightsquigarrow} agg(A)$ *a* ***refinement of*** P ***w.r.t.*** ϕ *if* $F' \supseteq F$*.*

In other words, a refinement is a pattern that describes a more specific trend, i.e., it subdivides each fragment $f \in frag(R, P)$ into one or more subsets based on the values of $F' - F$. Specifically, we are interested in drilling down into fragment $f = t[F]$ by exploring the data for $f' \in frag(R, P')$ where $f'[F] = t[F]$. Notice that F' does not have to be a subset of G, since more detailed aggregate results may also explain the user question. Besides, M' does not have to be the same as M, e.g. the number of papers an author publishes per year may increase linearly (`Lin`) over the years while at the same time he/she may also publish roughly same number of paper in SIGKDD every year (`Const`).

Example 4. *Continuing with Example 3 consider:*

$$P_2 = [\mathsf{author}, \mathsf{venue}] : \mathsf{year} \overset{\mathsf{Const}}{\rightsquigarrow} \mathsf{count}(*)$$

This pattern is a refinement of the relevant pattern P_1 *w.r.t.* ϕ_{A_X} *since* $F_2 = \{author, venue\} \supseteq \{author\} = F_1$*. This refinement corresponds to a drill-down from total publications per year for* A_X *to number of publications per venue and year.*

Explanations. Given a refinement P' for a pattern P w.r.t. a user question ϕ, we "drill-down" by considering *refined* fragments $f' \in frag(P', R)$ where $t[F] = f'[F]$, i.e., local versions of the pattern P', which agree with the question's tuple t on the partition attributes of the pattern P. For such f' we then search for prediction attribute assignments for which the aggregation function result *deviates* in the opposite direction (as predicted by the regression function) with respect to the question. For instance, we search for unusually high values to explain an atypically low value.

Having introduced pattern relevance, deviation, refinements, and drill-down based on refinements, we are ready to formally define explanations. Recall that $g_{P,f}$ denotes the regression model based on which P is determined to hold locally on fragment $f \in frag(P, R)$.

Definition 7 (Explanation). *Given a relation* R*, a user question* $\phi = (Q, R, t, dir)$ *where* $Q = \gamma_{\mathsf{G},\mathsf{agg(A)}}(R)$*, and a set of ARPs* $\mathcal{P}$ *that globally hold for local thresholds* θ, δ *and global thresholds* λ, Δ*, a* ***candidate explanation for*** ϕ *is a triple*

$$E = (P, P', t')$$

where $P = [F] : V \rightsquigarrow agg(A)$ *and* $P' = [F'] : V \rightsquigarrow agg(A)$ *are two patterns, and* $t' \in \gamma_{\mathsf{F'\cup V},\mathsf{agg(A)}}(R)$ *is a tuple, such that all of the following conditions hold:*

(1) $P \in \mathcal{P}$ *is relevant for* ϕ*,*
(2) $P' \in \mathcal{P}$ *is a refinement of* P *w.r.t.* ϕ*,*

(3) $t'[F'] \models_{\theta,\delta} P'$,
(4) $t'[F] = t[F]$ *and* $t' \neq t$ *(if from the same schema)*
(5) $dir = low \Rightarrow t'[agg(A)] > g_{P',t'[F']}(t'[V])$
$dir = high \Rightarrow t'[agg(A)] < g_{P',t'[F']}(t'[V])$

Note that we only require that P' holds locally for the counterbalance tuple t', but not the user question tuple t.

Example 5. *Continuing with Example 4, we will explain how to drill down to find explanations using the refinement P_2 of P_1 (author and venue as partition attributes refine author as partition attribute). For that, we first run the aggregate query $Q^* = \sigma_{author=A_X}(\gamma_{\{\text{author,venue}\}\cup\{\text{year}\},\text{count}(*)}(R))$; then we enumerate result tuples from this query to look for all combinations of author A_X with a venue (these are the refinements of $f = t[\{author\}] = A_X$ to $f' \in frag(P_2, Pub)$), and a year in the domain of the year attribute. For instance, using the data from Figure 2 for $v' = 2007$ and $f' = (A_X, ICDE)$, we compare the number of publications for the group 2007 ($t' = (A_X, ICDE, 2007) \in Q^*(R), t'[F_2] = f', t'[V_2] = v'$) with the number of publications predicted using the regression model ($g_{P_2,f'}(2007)$) that was fitted to determine that P_2 holds locally for f'. Assume that we obtained $g_{P_2,f'}(2007) = 1.2$, whereas $t'[agg(A)] = 2$. In this case the number of A_X's publications in ICDE in 2007 is higher than predicted (it deviates in the opposite direction of the user question). Thus, $E = (P_1, P_2, t')$ is one possible explanation for the user question. Intuitively, it can be interpreted as follows:*

Even though A_X like many other authors publishes roughly the same number of publications every year (pattern P_1 holds locally for A_X), his number of SIGKDD publications in 2007 is low which may be explained by his higher than usual number of ICDE publications in 2007 (P_2 holds locally for (A_X,ICDE), but year = 2007 is higher than predicted).

Given this explanation, a user may infer that in 2007 A_X possibly submitted some papers to ICDE instead of SIGKDD, and this might be a reason why the number of his SIGKDD publications is lower than other years.

To understand why both relevant patterns and refinements are needed to generate explanations, consider how our approach would be affected if we would only consider one of the two. If only relevant patterns are used, then counterbalances would have to be at least as coarse-grained as the user question which may exclude useful more specialized explanations. If we would only use refinements then there is no guarantee that the explanation has a connection with a pattern (trend) that holds for the user question.

3.3 Scoring Explanations

For an input dataset of realistic size and for a reasonably large number of patterns, the number of candidate explanations may be very large. Thus, showing all valid explanations to the user would be overwhelming. Furthermore, explanations that involve small deviations or where t' is quite dissimilar to the user question may not be meaningful. Therefore, we define a scoring function for explanations that determines an explanation's score based on (i) the similarity between the user question tuple t and the tuple t', and (ii) the amount $t'[agg(A)]$ deviates from the predicted value $g_{P',t'[F']}(t'[V])$. The rationale of (i) is that, explanations that are more similar to the user question are more likely to have caused the unusual result. For instance, a high number of publications in 2006 (or, a high number of publications in ICDE 2007) is more likely to be the cause of A_X's low number of SIGKDD publications in 2007 than a high number of publications in 2000 (or, in a systems conference, say SSDBM 2004). The motivation for (ii) is that a higher deviation from a predicted value indicates how unusual an event is and, intuitively, the more atypical an outcome is, the more likely it is to lead to other atypical events. For instance, if A_X published 35 papers instead of our predicted number of 35.47 in 2006, then this small deviation is not likely to affect his SIGKDD publications in 2007. Note that the scoring function we introduce here is only one possible option. Our approach is agnostic to what scoring function is used and new scoring functions can be added with ease. We give the formal definition of *deviation* below.

Definition 8 (Deviation). *Given a pattern $P = [F] : V \rightsquigarrow agg(A)$, and a tuple $t \in \gamma_{\text{F}\cup\text{V},\text{agg(A)}}(R)$ where P holds locally on t[F], the* ***deviation of t w.r.t. P*** *is defined as:*

$$dev_P(t) = t[agg(A)] - g_{P,t[F]}(t[V])$$

If $dev_P(t) > 0$ (resp. $dev_P(t) < 0$) then we say tuple t deviates positively (resp. negatively) w.r.t. P.

We compute the *distance* of two tuples as the L_2-norm of the weighted distances of their attribute values. We assume that weights for individual attributes and appropriate distance functions are given. We provide sensible defaults for distance functions (an attribute's domain is partitioned into classes – two values within the same class have low distance, values from different classes have higher distance, and the distance is 0 if the values are the same) and weights (equal weights for all attributes) as part of the implementation of explanation generation in our Cape system. Note that for the sake of generating a score, we will have to compare the tuple in the user question with an explanation tuple. However, the schemas of these tuples may not be the same. We address the problem by setting the attribute distance to 1 (maximal possible distance) for attributes that only appear in one of the two tuples. It is easy to incorporate other possible distance and weight functions in our framework.

Definition 9 (Distance). *Let Dom(A) denote the domain of attribute A, $w_A \in [0, 1]$ be the weight for attribute A such* *that $\sum_{A \in R} w_A = 1$, and $d_A : \textsc{Dom}(A) \rightarrow [0,1]$ denote a distance function (symmetric, and indiscernible values are identical with distance 0). Consider two tuples t_1 and t_2 with schemas T_1 and T_2. The distance of t_1 and t_2 is defined as:*

$$d(t_1, t_2) = \sqrt{\frac{1}{W} \sum_{A \in (T_1 \cup T_2)} w_A \cdot d_A^{exists}(t_1[A], t_2[A])^2}$$

$$d_A^{exists}(t_1, t_2) = \begin{cases} d_A(t_1[A], t_2[A]) & \text{if } A \in T_1 \cap T_2 \\ 1 & \text{otherwise} \end{cases}$$

$$W = \sum_{A \in (T_1 \cup T_2)} w_A$$

Recall that we require that the weights for attributes in R sum up to 1. Since $T_1 \cup T_2$ may not contain all attributes from R, we introduce a normalization factor $W = \sum_{A \in (T_1 \cup T_2)} w_A$ to make distances of tuples with different schemas (subsets of R) comparable.

The score we assign to an explanation is computed as the deviation divided by the distance and further normalized using the value of the predictor attribute of the relevant pattern from where we initiated the drill-down. The rationale for this method is that higher deviation is desirable as is lower distance. Furthermore, the motivation for the normalization factor is that the relative value of the deviation has to be put into context. For instance, if A_X had published a total of 10 papers in 2007, then if the actual number of publications in ICDE 2007 were 3 higher than expected, it would have had a much higher impact than the actual scenario where A_X published 48 papers in 2007.

Definition 10 (Score). *Let $\phi = (Q, R, t, dir)$ be a user question where $Q = \gamma_{\mathrm{G,agg(A)}}(R)$, and $E = (P, P', t')$ for $P = [F] : V \rightsquigarrow agg(A)$ and $P' = [F'] : V \rightsquigarrow agg(A)$ be a candidate explanation for ϕ. The score $score(E)$ is computed as:*

$$score(E) = \frac{dev_{P'}(t') \cdot isLow}{d(t[G], t'[F', V]) \cdot \textsc{norm}} \qquad (1)$$

where $\textsc{norm} = \pi_{agg(A)}(\sigma_{F=t[F] \wedge V=t[V]}(\gamma_{\mathrm{F \cup V, agg(A)}}(R)))$ *and* $isLow = 1$ *if* $dir = low$ *and* -1 *otherwise*[2].

For P', t' originating from the refinement of multiple original patterns P, we keep the $E = (P, P', t')$ with the highest score. The normalization factor helps to remove non-influential candidate explanations by assigning them a very low score. For instance, assume that SIGKDD accepts about the same number of papers every year and a pattern $P_3 = [venue] : year \overset{\mathrm{Const}}{\rightsquigarrow} count(*)$ with $F = \{venue\}$ and $V = \{year\}$ holds globally and also locally for $f = SIGKDD$. The refinement of pattern P_3 will also generate the pattern P_2 with $F = \{author, venue\}$ and $V = \{year\}$. The candidate explanations originating from this refinement will involve SIGKDD publications of other authors ($\neq A_X$) in different years. However, the normalization factor in the denominator, $\textsc{norm}$, will be the total number of papers accepted in SIGKDD

[2]A small ϵ is added to the denominator to avoid division by zero if $\textsc{norm} = 0$.

Algorithm 1 Find explanations for ϕ using ARPs $\mathcal{P}$

```
Naive-Generate-Expl($\phi = (Q, R, t, dir), \mathcal{P}, (\theta, \delta), (\lambda, \Delta), k$)
 1  $\mathcal{E} = \emptyset$ // min-heap: top-$k$ explanations sorted on $score$
 2  for $P = (F, V, agg, A, M) \in \mathcal{P}$
 3      if $F \subseteq G \wedge t[F] \models_{\theta,\delta} P$ // Pattern is relevant?
 4          $\mathcal{P}_{refined}$ = getRefinements($P, \mathcal{P}$)
 5          for $P' = (F', V, agg, A, M) \in \mathcal{P}_{refined}$
 6              for $t' \in \gamma_{\mathrm{F' \cup V, agg(A)}}(R)$
 7                  if $t'[F] = t[F] \wedge t'[F'] \models_{\theta,\delta} P'$
 8                      // Update min-heap
 9                      updateExpl($\phi, \mathcal{E}, t'P, P'$)
10  return $\mathcal{E}$

updateExpl($\phi = (Q, R, t, dir), \mathcal{E}, t', P, P'$)
1  if $(dir = low \wedge t'[agg(A)] > g_{P', t'[F']}(t'[V]))$
       $\vee(dir = high \wedge t'[agg(A)] < g_{P', t'[F']}(t'[V]))$
2      $E = (P, P', t')$
3      if (numElements($\mathcal{E}$) $< k$)
4          insertHeap($\mathcal{E}, E$)
5      elseif ($score$(Peek($\mathcal{E}$)) $< score(E)$)
6          removeRoot($\mathcal{E}$)
7          insertHeap($\mathcal{E}, E$)

getRefinements($\phi, P, \mathcal{P}$)
1  $\mathcal{P}_{refined} = \emptyset$
2  for $P' = (F', V', agg', A', M') \in \mathcal{P}$
3      if $F' \supseteq F \wedge V' = V \wedge agg' = agg \wedge A' = A$
4          $\mathcal{P}_{refined} = \mathcal{P}_{refined} \cup \{P'\}$
5  return $\mathcal{P}_{refined}$
```

2007 across all authors, and since this number is high, the score of individual explanations will be low, validating the intuition that a higher or lower number of publications of other authors possibly does not affect the lower publication count of A_X in SIGKDD 2007.

3.4 Generating and Ranking Explanations

We now introduce a brute-force algorithm for generating explanations (Algorithm 1) and discuss optimizations later. The algorithm takes as input a set of ARPs $\mathcal{P}$ and a user question $\phi = (Q, R, t, dir)$. We first initialize a min-heap sorted on the scores of explanations that is used to hold the top-k explanations we have found so far. The algorithm then iterates over all patterns in $\mathcal{P}$. For each pattern $P \in \mathcal{P}$, it first checks whether the pattern is relevant (Definition 5) for question ϕ (line 3). For each relevant pattern P we then explore all drill-down options (lines 5 to 9). That is, we iterate over all patterns P' that are refinements of P and all tuples t' over schema $(F', V, agg(A))$, and for each such tuple check whether it is an explanation (procedure updateExpl). For each explanation, we first check if we have generated k explanations yet (line 3). If this is not the case then we insert the explanation into the heap; otherwise, we check whether its score is higher than the score of the heap's root element (which has the lowest score among the top-k explanations we have found so far) and update the min-heap by removing the root element and inserting the explanation (line 5).

3.5 Upper Score Bound and Optimizations

A disadvantage of the brute force algorithm is that we need to check all candidate explanations, even if it is guaranteed that they will not be part of the top-k explanations. To enable pruning of the search space we introduce an upper bound on the score of explanations and use this upper bound to prune sets of candidate explanations whose upper bound is lower than the lowest score of the top-k explanations we have found so far, i.e., that are guaranteed to not be part of the result. Furthermore, we keep a priority queue of candidates that allows us to efficiently prune explanation candidate sets.

Consider the formula for computing the score of an explanation $E = (P, P', t')$ given in equation (1). Recall that NORM is computed based on P and the user question alone. Thus, this term will be the same for any refinement of a pattern P according to a user question. If we can find a *lower bound for distance d* and an *upper bound for deviation dev*, then this gives us an upper bound for *score*. Specifically, we are interested in bounds for any t' given P and P' because this would allow us to avoid generating any candidate explanations for patterns P' where the upper score bound for any explanation involving P' as refinement is lower than the minimum top-k score we have found so far. Furthermore, this will help us heuristically prioritize patterns with higher bounds with the hope to detect better explanations early-on and, thus, also prune more subsequent candidates.

Recall that $t'[F] = t[F]$ by construction. Let T, T' be the attributes of t, t'. Based on the definition of d, the distance of attributes that are not in $T \cap T'$ is set to the maximum value of 1. The lowest distance value that could be achieved for a pattern P' under these constraints is when the distance of all of the remaining attributes is 0. Thus, we get an lower bound $d_{\downarrow}(\phi, P') = |F' - G|$ for the distance for any t' for a refined pattern P' and the user question tuple t.

For the deviation we can simply compute the maximum positive and negative deviation across all $f \in frag(P, R)$ and $v \in \textsc{Dom}(V)$ and store this information with the pattern (this can be integrated with our pattern mining algorithm without adding significant overhead).

$$dev_{\uparrow}(\phi, P) = \begin{cases} \max_{t \in \gamma_{\mathsf{F \cup V, agg(A)}}(R)}(dev_P(t)) & \text{if } dir = low \\ \min_{t \in \gamma_{\mathsf{F \cup V, agg(A)}}(R)}(dev_P(t)) & \text{if } dir = high \end{cases}$$

Combining these bounds we get an upper bound for any explanation for a user question ϕ which is based on a relevant pattern P being refined to pattern P':

$$score_{\uparrow}(\phi, P, P') = score(E) = \frac{dev_{\uparrow}(\phi, P')}{d_{\downarrow}(\phi, P') \cdot \textsc{norm}}$$

We amend Algorithm 1 to incorporate these optimizations. First, we iterate over patterns P in decreasing order of the normalization factor in the denominator NORM. Then we compute $score_{\uparrow}(\phi, P, P')$ to decide whether to drill down from this pattern. If the score bound is lower than the lowest value of the current set of top-k tuples, this pattern is skipped. Similarly, if P is not pruned, then for each pattern refinement P' of P, we decide whether to check all candidate explanation tuples by computing a more accurate bound using the information stored with the local versions of a pattern. We do not show the modified pseudocode here since the modifications are straightforward.

4 PATTERN MINING

We now present our solution for mining ARPs for an input relation R. We split the task of pattern mining into two subtasks: (i) enumeration of candidate patterns, and (ii) checking whether a pattern candidate holds globally. We first introduce a naive algorithm as a baseline and then present several optimizations for the two subtasks. Here we do not try to optimize regression analysis needed for generating a model and for calculating its goodness-of-fit (GoF). We simply treat regression analysis as a blackbox to which we feed data as input, and get back a model and the GoF value.

Brute-force algorithm for pattern discovery. The naive way of candidate enumeration is to generate all possible combinations (F, V, agg, A, M) for any $F \subset R$, $V \subset R$, $A \in R$ such that $F \cap V = \emptyset$ and $A \notin (F \cup V)$, aggregation function agg, and regression model types M. Then we can determine whether a particular pattern candidate $P = (F, V, agg, A, M)$ holds globally over R by retrieving the relevant data and running a regression. The number of patterns considered by this brute-force method is exponential in the number of attributes of the relation R. Although the runtime is polynomial if we consider data complexity [45] (the number of attributes is constant), for relations with more than a few attributes, this method is not efficient. We give the pseudocode of the brute-force method in Appendix C (Algorithms 3 and 4). To develop a more efficient approach, we introduce several optimizations in the following subsections. Some of these optimizations do not affect the output of the algorithm while others are heuristic in nature, i.e., we may not find all patterns.

4.1 Optimizations

Restricting pattern size. Consider the group-by attributes G of the aggregate query of a user question. Typically, we can expect $|G|$ to be small. Explanations for a question are unlikely to use patterns where $F \cup V$ is much larger than G, because such patterns would be much more fine-grained than the user question. Hence, we only consider patterns for which $|F \cup V| \leq \psi$ for a configurable threshold ψ. This threshold can either be set conservatively, or, we can start with a lower threshold and rerun pattern mining with a larger threshold if a user question with a large $|G|$ is asked. This reduces the number of patterns from exponential in $|R|$ to exponential in ψ.

One query for all patterns sharing F and V. To check whether a pattern P holds globally, we check whether P holds locally for a sufficiently large number of fragments. Instead of retrieving the tuples for each fragment using a separate query, we can run a single query with $F \cup V$ as the group-by attributes, and order the result first on F and then on V to retrieve all fragments at once. Furthermore, this query can be extended to evaluate all possible combinations $agg(A)$ of an aggregation function $agg \in \mathcal{A}$ and input attribute $A \in R$ at the same time. Thus, a single query is sufficient to test all patterns sharing the same F and V.

One query per $F \cup V$. Consider a set of pattern candidates that share the same $F \cup V$. The group-by queries that we run for patterns in this set only differ in the sort order they use. Thus, instead of running a group-by query for each such pattern, we can materialize the result of a single group-by query for each set $F \cup V$, and then sort the results for each particular F and V. These sort queries are often faster than the aggregation queries, since the result of aggregation is typically significantly smaller than R.

Using the `CUBE BY` operator. Most SQL databases support the cube operator [16], which evaluates an aggregation query over multiple sets of group-by attributes returning the union of all these group-by queries. Given a set of group-by attributes G, the cube operator computes aggregations over all subsets of G. Thus, we can utilize the cube operator to compute the data required for all pattern candidates in a single query. In the following we write G_P to denote $F \cup V$ for a pattern P. Since we are only interested in patterns where $|G_P| \leq \psi$, computing the cube over R would return some groupings which do not correspond to a valid pattern candidate. We use the SQL `GROUPING(a)` construct to filter out invalid groups (details omitted due to space constraints). We materialize the output of this query. To retrieve the data for one pattern P, we evaluate a selection and order-by query over the materialized result. The number of such queries we have to evaluate over the materialized result is the same as that for the previous optimizations. Even though most DBMSs optimize the execution of the cube operator, the exponential number of groups that are generated results in poor performance for relations with large number of attributes.

Reusing sort orders. Ideally, we would like to avoid having to re-sort the result of a query grouping on a set G to fit the sorting requirements for a pattern P. We make two observations: (1) there are multiple sort orders that fulfill the requirement that the partition attributes F should be a prefix of the sort order giving us the flexibility to choose the one we deem most beneficial, and (2) if $F_1 \supset F_2$ for two patterns P_1 and P_2 where $(F_1 \cup V_1) = (F_2 \cup V_2)$ then we can reuse the sort order for P_1 to evaluate P_2 as long as the attributes of F_2 form a prefix of the sort order chosen for P_1. For instance, if $F_1 = \{A, B, C\}$ and $F_2 = \{A, C\}$ and $V_1 = \{D\}$ and $V_2 = \{B, D\}$ then a sort order (A, C, B, D) can be used for both P_1 and P_2. Our optimized algorithm discussed in Section 4.2 exploits this observation to reduce the number of sort queries we have to run.

Algorithm 2 ARP-mine pattern discovery

```
ARP-MINE($R, \Psi, \psi, \theta, \delta, \lambda, \Delta$)
1  $\mathcal{P} = \emptyset$            // Patters that hold globally
2  $C = \emptyset$            // Candidate $(F, V)$ we have considered
3  $groupSizes = \emptyset$   // Map of $G$ to $|\pi_G(R)|$
4  for $i \in \{2, \dots, \psi\}$
5      for each $G \subset R$ and $|G| = i$
6          $A_{agg} = R - G$
           // for all $A_i \in A_{agg}, agg \in \mathcal{A}$
7          $Q$ = SELECT G, sum($A_1$), ... FROM R GROUP BY G
8          $D = Q(R)$
9          INSERT($groupSizes, G, |D|$)
10         $\Psi$ = DETECTFDs($groupSizes, G, \Psi$)
11         $(\mathcal{P}, C)$ = EXPLORESORTORDER($G, D, C, \mathcal{P}, R, \theta, \delta, \lambda, \Delta$)
```

4.2 The ARP-mine Algorithm

We now present an enhanced algorithm that integrates the optimizations we have presented so far. Algorithm 2 takes as input a relation R, a set of functional dependencies[3] Ψ, a pattern size threshold ψ, and thresholds $(\theta, \delta), (\lambda, \Delta)$. It returns a set of patterns that hold globally with $(\theta, \delta), (\lambda, \Delta)$ and where $|G_P| \leq \psi$. The algorithm uses two variables to keep track of patterns that are known to hold globally ($\mathcal{P}$) and the pattern candidates that have been evaluated so far (C). The algorithm explores pattern attribute sets in increasing size from the minimal possible size of 2 (both F and V consist of a single attribute) up to the threshold ψ. We enumerate candidate patterns in increasing size of $G_P = F \cup V$, because this allows us to detect functional dependencies (FDs) $A \longrightarrow B$ from the data, where B is a singleton attribute, to evaluate patterns with $G_P = A \cup \{B\}$. The rationale for this is that, as we show in Appendix D, we can skip patterns where an attribute A in F is implied by $F - \{A\}$. Since patterns have at least one predictor attribute (V), FDs of the form $(F - \{A\}) \longrightarrow A$ would have been discovered in an iteration before the current iteration considers patterns with F, i.e., the FDs needed to skip a pattern will have been discovered before the pattern is considered.

[3]This initial set of FDs can include FDs provided by the user and/or FDs determined based on primary keys and uniqueness constraints that, for most database systems, can be determined by querying the system catalog.

For each size i (line 4), we generate all subsets G of R with cardinality i. For each subset G we compute a query grouping on G to evaluate all combinations of aggregation functions *agg* and input attributes A. We record the number of groups in the result of this query in a map *groupSizes* and then use this map to check whether FDs $(G - \{A\}) \rightarrow A$ hold for some $A \in G$ (as described in Appendix D). Afterwards, procedure ExploreSortOrders (Algorithm 5) is called to test all pattern candidates where $G_P = G$.

Procedure ExploreSortOrders generates all permutations S of G (line 1). Before running the sort query which sorts the output of the aggregation query D according to S, we first check whether there is at least one (F, V) combination which we have not considered so far and that can be tested using S (line 2). Note that we use $\mathcal{C}$ to store all (F, V) pairs that we have explored so far. If no F, V pair exists that can be tested based on S and has not been tested already, then we can skip S. Otherwise, we generate all pairs (F, V) where $F \cup V = G$ such that a permutation of F is a prefix of the sort order S (line 5). For each such pair we add it to $\mathcal{C}$. We then check whether it has been checked before and whether patterns with $G_P = (F \cup V)$ can be skipped based on the FDs that hold over the input R (as described in Appendix D). If a pair passes these checks then we generate all candidates for G_P by choosing an aggregation function and input attribute (or $*$ if $agg = count$). Each candidate is tested using FitPattern (Algorithm 6, Appendix C).

This algorithm scans through the sorted aggregation result (D_{sort}) processing one $f \in frag(P, R)$ at a time. Since F is a prefix of the sort order S, we know that all tuples t with $t[F] = f$ will occur as one consecutive block in the result. Thus, we can process the data for one f at a time by constructing the mapping from V to the aggregation result and then use regression to evaluate whether the pattern holds locally for f. Once we have scanned through D_{sort} we have explored all $f \in frag(P, R)$ and can now determine whether the pattern holds globally and return either true or false. Note that for simplicity we have presented Algorithm 6 to evaluate a single pattern. To reduce the number of scans of D_{sort} we evaluate all pattern candidates for S using one scan of D_{sort} (essentially evaluating multiple patterns in parallel).

5 EXPERIMENTS

In this section, we evaluate the performance of explanation generation algorithms presented in Section 3, the performance of the aggregate regression patterns mining algorithms presented in Section 4, and the quality of explanations generated by our approach. We use two real world datasets in the experiments. A qualitative analysis and user study are presented in Appendix A and B, respectively.

DBLP. *DBLP* is a bibliography dataset extracted from DBLP (http://dblp.uni-trier.de/) which consists of a table `Pub(author, pubid, year, venue)` recording authors and their publications. For each publication we record the venue, publication year, and a unique identifier `pubid`. We created versions of this dataset ranging from 10k to 1M rows.

Chicago crime. The `Crime` dataset (https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-present/) reports crimes in Chicago from 2001 to 2017. It contains 6.5M rows and has 22 attributes. We dropped attributes whose values are almost unique (e.g., `case_id`, `latitude`, `longitude`), and we split some attributes into multiple parts (month, week, etc.). The remaining attributes (e.g., `block`) are discrete with a domain size between 2 and 59k. Based on this preprocessing step, we created versions of varying size (10k to 1M rows) and number of attributes (4 to 11).

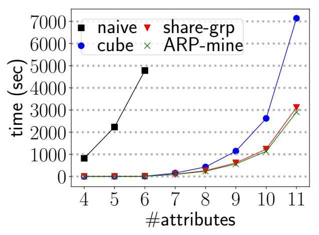
Experimental setup. Cape is implemented in Python (version 3.5) and runs on top of PostgreSQL (version 10.4). All experiments were run on a machine with 2 x AMD Opteron 4238 CPUs, 128GB RAM, and 4 x 1 TB 7.2K RPM HDDs in hardware RAID 5. The Cape framework consists of two main steps: (i) Mine ARP offline, and (ii) find the top-k explanations for a user question. We evaluate these two steps separately.

Parameters. We vary the following parameters: (1) D: the number of rows in the input dataset; (2) A: the number of attributes in the input schema; (3) ψ: the maximal size of G_P considered (see Section 4); and (4) thresholds $(\theta, \lambda, \delta, \Delta)$. For explanation generation, we also vary (5) A_ϕ: the number of group-by attributes in the user question and (6) N_P: the number of local patterns by using only a subset of the patterns detected during explanation generation.

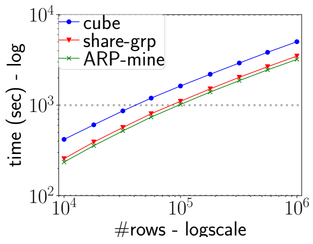
5.1 Mining ARPs

We first evaluate variants of the ARP mining algorithm introduced in Section 4. Based on preliminary experiments we found that the variation in runtime for repeated executions of the same ARP-mining task is negligible (less than 2%) and, thus do not report variance here. For this experiment, we consider Naive (the brute force algorithm), Cube (which uses a single cube aggregation query), Share-grp (which shares group-by queries for all pattern with the same group-by attributes), and ARP-mine (which shares group-by queries and sort orders (Algorithm 2)). We set $\psi = 4$, $\theta = 0.5$, $\lambda = 0.5$, $\delta = 15$, and $\Delta = 15$. We deactivate the FD optimizations presented in Appendix D.

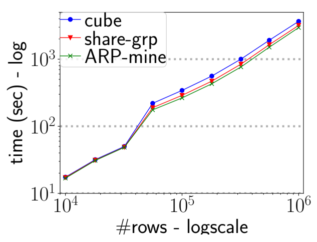
Crime dataset. Figure 3a shows the runtime for pattern mining for the Crime dataset varying the number of attributes (A). Since we set $\psi = 4$, the number of pattern candidates that needs to be considered and, thus, also the runtime is $O(A^4)$. Naive already takes 18,000 seconds for 7 attributes (data point omitted to increase legibility) demonstrating the need



(a) Crime, $D = 10k$, varying A



(b) Crime, $A = 8$, varying D



(c) DBLP, $A = 4$, varying D

Figure 3: Measuring ARP mining runtime for ARP-MINE, SHARE-GRP, and CUBE

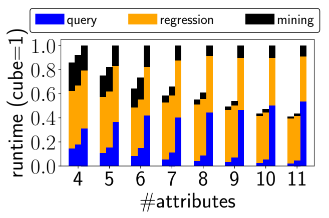


Figure 4: Subtask runtime, Crime, $D = 10k$, varying A

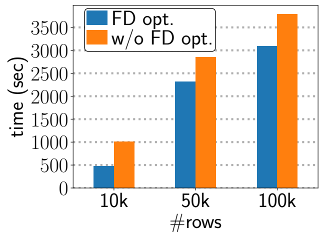
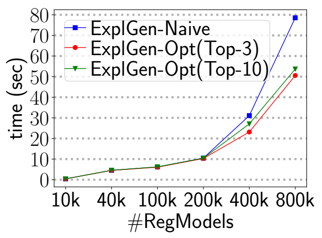
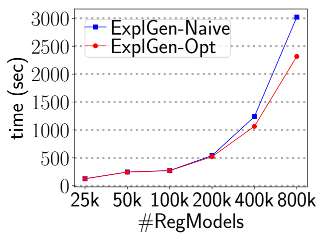


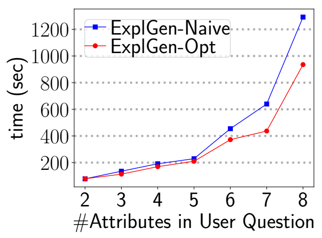
Figure 5: FD optimizations, Crime, $A = 9$, varying D



(a) DBLP, $D = 5M$, $A = 4$



(b) Crime, $D = 1M$, $A = 8$



(c) Crime, $D = 1M$, $A = 8$

Figure 6: Measuring explanation generation runtime

for optimization. ARP-MINE slightly outperforms SHARE-GRP (~ 7%). Both ARP-MINE and SHARE-GRP outperform CUBE with a margin that increases in A (~ 16% for $A = 4$ up to ~ 145% for $A = 11$). This is not surprising, because the number of groups the `CUBE` operator has to generate is exponential in A.

Figure 3b shows the result for varying D (NAIVE is omitted). The runtime of all methods increases linearly in D. Again this expected, since both aggregation queries and regression analysis are linear in D. As expected, ARP-MINE outperforms the other methods. SHARE-GRP exhibits 8% (9%) overhead compared to ARP-MINE for $D = 10k$ ($D = 1M$). CUBE exhibits 78% (57%) overhead for $D = 10k$ ($D = 1M$).

DBLP dataset. Figure 3c shows the result for varying D for the DBLP dataset. We fix $A = 4$ since this dataset has only 4 attributes. Again the runtime is linear in D and ARP-MINE performs best. The performance difference between the methods is less pronounced than for the Crime data which is explained by the low number of attributes (cf. Figure 3a).

Subtask performance. To better understand the observed behavior we drill-down into the individual subtasks. Figure 4 shows the runtime of regression, query processing, and the remaining mining tasks. Within each group of bars the left bar is ARP-MINE, the middle one SHARE-GRP, and the right one CUBE. We use the crime dataset ($D = 10k$) and vary A. The runtime of the different methods is normalized to the slowest method (CUBE). As expected all methods spend the same amount of time on regression. With increasing number of attributes the fraction of the runtime spend on regression increases significantly. The exception is CUBE for which the time spend on query processing increases from ~ 30% at 4 attribute to ~ 50% at 11 attributes.

Functional dependencies. We use the Crime dataset with 9 attributes ($A = 9$, varying D) for which a reasonable amount of FDs holds (note that this set of 9 attributes is different from the set used in Figure 3a) and compare our ARP-MINE with and without FD optimizations activated (Appendix D). Activating the FD optimizations (Figure 5) has a positive effect on runtime (improvement of ~18% to ~53%). In addition to improving performance, the FD optimizations also prune spurious patterns (see Appendix D).

5.2 Finding Explanations - Performance

We now evaluate the performance of explanation generation. The number of local patterns have a significant effect on performance. For this experiment we run pattern mining offline to generate a large number of patterns. We then generate explanations based on subsets of these patterns controlling the total number of local patterns (N_P). We compare EXPLGEN-NAIVE which is the brute force algorithm (Section 3.4) with EXPLGEN-OPT which is our optimized algorithm that prunes parts of the search space (Section 3.5).

Generating user questions. We generate aggregate queries by picking 2 to 8 group-by attributes for the Crime dataset (2 to 4 for the DBLP dataset). For each query, we create several user questions by randomly selecting result tuples. We bias the selection to prefer groups with large counts to create a worst case scenario for explanation generation.

DBLP dataset. For the DBLP dataset we set $D = 5M$ and use all 4 attributes ($A = 4$). Figure 6a shows the runtime of

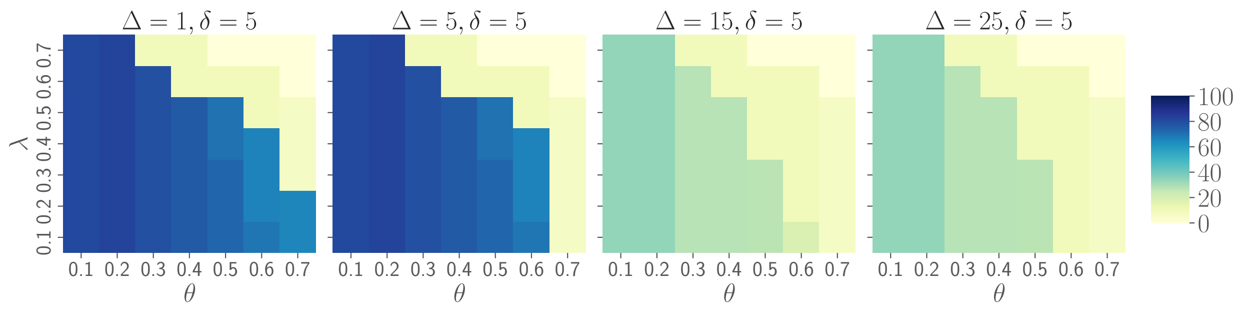


Figure 7: Parameter sensitivity analysis - measuring precision w.r.t. to ground truth for varying $(\theta, \delta), (\lambda, \Delta)$

explanation generation when varying the number of local patterns (N_P). The runtime can vary significantly based on the choice of UQ. We generated several user questions and report the total time for generating explanations for all of these questions. The runtime of explanation generation increases linearly in N_P. ExplGen-Opt outperforms ExplGen-Naive with the margin increasing in N_P (up to 35%).

Crime dataset. Figure 6b shows the results for the Crime dataset $D = 1M$. Again the runtime is linear in N_P and ExplGen-Opt outperforms ExplGen-Naive (up to 28%). We also evaluate the effect of the number of user question group-by attributes (parameter A_ϕ) on runtime (Figure 6c).

5.3 Parameter Sensitivity

To evaluate how the thresholds $(\theta, \delta), (\lambda, \Delta)$ affect the capability of our approach to return meaningful explanations we created datasets with known ground truth counterbalances and tested whether our approach is able to find the correct explanation under varying parameter settings. To generate ground truth explanations we started from the real-world datasets and considered the patterns mined for these datasets. To introduce an outlier and its counterbalance we randomly selected a pattern, one partition attribute value, and one predictor attribute value. We then increased or decreased the aggregation result by removing or adding tuples from/to the set of data points corresponding to this combination. For each outlier we introduced a counterbalance for different values of the partition and predictor attributes.

We generated 10 user questions and then the top-10 explanations using Cape. In Figure 7, we report the percentage of ground truth explanations among the 100 explanations (10×10) returned by Cape for various parameter settings. We vary the local model quality threshold θ, global support threshold Δ, and global confidence threshold λ. In general, the number of ground truth explanations found by Cape decreases when θ is increased; but if θ is too low, there will be explanations from patterns with low regression model quality resulting in the inclusion of incorrect explanations which in turn reduces precision if θ is lower than 0.2 for $\Delta \leq 5$. The global confidence threshold (λ) has less of an effect when θ is low, but the increase in λ will significantly reduce the number of patterns available for generating explanations when θ is high. The global support threshold Δ largely depends on the distribution of the data, and it will affect the number of patterns that hold globally: when Δ is 15 or 25, we find much fewer ground truth explanations. The local support threshold δ is used for avoiding meaningless patterns, like a linear regression model for a single data point. Based on our experience, as long as this parameter is within a reasonable range, it does not affect the precision significantly. In summary, we recommend to set Δ to a low value. For the global confidence we observe that choosing a value in the range between $0.3 - 0.5$ is reasonable unless θ is very high. Finally, lower values of θ are more effective. The reason is that even though not all patterns with lower goodness-of-fit are meaningful, the existence of these patterns does not significantly affect precision, because the ranking of explanations helps to filters out meaningless explanations.

6 RELATED WORK

Provenance. Data provenance for relational queries records how results of a query depend on the input data [8, 10, 12, 17]. Although provenance has primarily been studied for non-aggregate queries, the *provenance semiring* framework by Green et al. [17] was extended to aggregate queries by Amsterdamer et al. [6]. This approach replaces the data domain with symbolic expressions that combine elements of a semiring with elements from the domain of an aggregation function to record how an aggregation function result is computed from the input tuples and values. A simpler, but less expressive, model for provenance of aggregate queries has been explored in the context of the Perm and GProM systems [7, 15]. Summarization techniques have been used to compactly represent provenance (e.g, Ainy et al. [5] and Lee et al. [29]). However, as discussed earlier, provenance is not sufficient for the type of explanations we study here.

Missing answers and why-not questions. The problem of explaining missing query results has been studied using two approaches: instance-based [20, 21, 24, 28] where explanations are (missing) input tuples, and query-based [9, 44] where explanations are based on query predicates or operators. Ten Cate et al. [43] studied the problem of finding the most general explanations for why-not questions using ontologies. For aggregate queries, Meliou and Suciu [31] studied how to attain a desired change of a query answer by updating the database. In contrast, we explain outcomes based on patterns and counterbalance.

Explanations for query answers. Wu-Madden [47], Roy-Suciu [36], and Roy et al. [35] proposed frameworks for intervention-based explanations for outliers in aggregate query results. Given a user question (similar to ours), these methods find sets of tuples that, if removed, cause the answer to change in the direction opposite to the direction of the outlier. An explanation is then a compact summary of such tuples represented as a selection predicate. In contrast to our approach, the explanations produced by this line of work are restricted to *tuples in the provenance.*

Several other approaches have been proposed to explain query answers. They include *causality and responsibility* (measuring the degree of contribution of tuples in the provenance, e.g., [30]), and explanations for specific applications like hierarchical summaries [1], summarized explanation tables for a binary outcome [14], for probabilistic databases [26] and map-reduce job performance [27].

OLAP and data cube exploration techniques. Sarawagi and Sathe [37, 39, 40] proposed techniques for efficient data analysis over OLAP data cubes [16]. They proposed operators like *RELAX* (generalizes a specific problem case and returns all possible maximal contexts in which the problem occurs), *DIFF* (reports the main summarized differences between two values observed at aggregate levels), and *SURPRISE* (helps a user to quickly familiarize herself with the significant features of a OLAP cube) to automate the analysis process. Sarawagi et al. [38] studied a discovery-driven exploration approach to mine data for *exceptions* in patterns and lead the analyst to interesting regions of a data cube. More recently, Joglekar et al. [25] proposed the *smart drill-down* operator for interactive exploration and summarization of interesting (and possibly unexplored) tuples. These approaches help interactive exploration of data, but do not automatically generate explanations for user questions.

Data mining. Pattern detection has a long tradition in fields like machine learning, knowledge discovery, and natural language processing. Similar to the goal of many data mining methods [18] such as association rule mining [4, 22], ARPs compactly describe trends in data. However, our approach differs in that we mine such patterns over the results of aggregation queries. The algorithms for mining ARPs in this paper are motivated by the optimizations in the seminal work on association rule mining [4] and data cube [2, 19]. However, we needed to develop new techniques since the monotonicity properties used in these works do not hold in general for our problem. Outlier detection is also an active field (e.g., see these surveys [3, 23]), but simply detecting outliers is insufficient for explaining the underlying causes.

Interactive and visual data exploration. Visual data exploration such as Voyager 2 [46], Zenvisage [41], Vizdom [11], and Tableau (Polaris) [42] aid users in understanding characteristics of a dataset. Such techniques are very effective for detecting outliers and can potentially be useful for identifying simple ARPs. However, the large search space for counterbalances for an outlier and the required correlation between the outlier and user question provided by patterns make even a guided manual exploration infeasible. Thus, we consider these techniques as complementary to our approach. For example, they can be used to determine outliers and, thus, help the user identify meaningful questions or to visualize Cape's explanations.

7 CONCLUSIONS

Existing techniques for explaining outliers in aggregation results are often of limited applicability, because they only consider inputs that belong to the outlier's provenance. To overcome this shortcoming, we presented a novel framework for explaining outliers based on *counterbalances* w.r.t. patterns (ARPs) that hold over the data in aggregation. We developed efficient algorithms for mining ARPs and for generating explanations using ARPs. In future work, we plan to support more diverse types of queries and develop a unified system that combines explanations through counterbalance with explanations through generalization/specialization and provenance. Another challenging open research question is how to deal with missing values in user queries, patterns, and in explanations (e.g., if A_X did not have any SIGKDD paper in 2007).

ACKNOWLEDGMENTS

We thank Oliver Kennedy for useful discussions in the early stage of this work. This work is supported in part by NSF Awards IIS-1552538, IIS-1703431, OAC-1541450, OAC-1640864, SMA-1637155, and NIH Award 1R01EB025021-01.

REFERENCES

[1] Deepak Agarwal, Dhiman Barman, Dimitrios Gunopulos, Neal E. Young, Flip Korn, and Divesh Srivastava. 2007. Efficient and Effective Explanation of Change in Hierarchical Summaries. In *KDD*. 6–15.

[2] Sameet Agarwal, Rakesh Agrawal, Prasad Deshpande, Ashish Gupta, Jeffrey F. Naughton, Raghu Ramakrishnan, and Sunita Sarawagi. 1996. On the Computation of Multidimensional Aggregates. In *VLDB*. 506–521.

[3] Charu C Aggarwal and Philip S Yu. 2001. Outlier detection for high dimensional data. In *ACM Sigmod Record*, Vol. 30. 37–46.

[4] Rakesh Agrawal and Ramakrishnan Srikant. 1994. Fast Algorithms for Mining Association Rules in Large Databases. In *VLDB*. 487–499.

[5] Eleanor Ainy, Pierre Bourhis, Susan B. Davidson, Daniel Deutch, and Tova Milo. 2015. Approximated Summarization of Data Provenance. In *CIKM*. 483–492.

[6] Yael Amsterdamer, Daniel Deutch, and Val Tannen. 2011. Provenance for Aggregate Queries. In *PODS*. 153–164.

[7] Bahareh Sadat Arab, Su Feng, Boris Glavic, Seokki Lee, Xing Niu, and Qitian Zeng. 2018. GProM - A Swiss Army Knife for Your Provenance Needs. *IEEE Data Eng. Bull.* 41, 1 (2018), 51–62.

[8] Peter Buneman, Sanjeev Khanna, and Wang Chiew Tan. 2001. Why and Where: A Characterization of Data Provenance. In *ICDT*. 316–330.

[9] Adriane Chapman and H. V. Jagadish. 2009. Why not?. In *SIGMOD*. 523–534.

[10] James Cheney, Laura Chiticariu, and Wang Chiew Tan. 2009. Provenance in Databases: Why, How, and Where. *Foundations and Trends in Databases* 1, 4 (2009), 379–474.

[11] Andrew Crotty, Alex Galakatos, Emanuel Zgraggen, Carsten Binnig, and Tim Kraska. 2015. Vizdom: Interactive Analytics through Pen and Touch. *PVLDB* 8, 12 (2015), 2024–2027.

[12] Yingwei Cui, Jennifer Widom, and Janet L. Wiener. 2000. Tracing the lineage of view data in a warehousing environment. *ACM Trans. Database Syst.* 25, 2 (2000), 179–227.

[13] Norman R Draper and Harry Smith. 2014. *Applied regression analysis*. Vol. 326. John Wiley & Sons.

[14] Kareem El Gebaly, Parag Agrawal, Lukasz Golab, Flip Korn, and Divesh Srivastava. 2014. Interpretable and Informative Explanations of Outcomes. *PVLDB* 8, 1 (2014), 61–72.

[15] Boris Glavic, Renée J. Miller, and Gustavo Alonso. 2013. Using SQL for Efficient Generation and Querying of Provenance Information. In *In Search of Elegance in the Theory and Practice of Computation - Essays Dedicated to Peter Buneman*. 291–320.

[16] Jim Gray, Surajit Chaudhuri, Adam Bosworth, Andrew Layman, Don Reichart, Murali Venkatrao, Frank Pellow, and Hamid Pirahesh. 1997. Data Cube: A Relational Aggregation Operator Generalizing Group-by, Cross-Tab, and Sub Totals. *Data Min. Knowl. Discov.* 1, 1 (1997), 29–53.

[17] Todd J. Green, Grigoris Karvounarakis, and Val Tannen. 2007. Provenance semirings. In *PODS*. 31–40.

[18] J. Han and M. Kamber. 2001. *Data Mining: Concepts and Techniques*. Morgan Kaufmann.

[19] Venky Harinarayan, Anand Rajaraman, and Jeffrey D. Ullman. 1996. Implementing Data Cubes Efficiently. In *SIGMOD*. 205–216.

[20] Melanie Herschel and Mauricio A. Hernández. 2010. Explaining Missing Answers to SPJUA Queries. *PVLDB* 3, 1 (2010), 185–196.

[21] Melanie Herschel, Mauricio A. Hernández, and Wang Chiew Tan. 2009. Artemis: A System for Analyzing Missing Answers. *PVLDB* 2, 2 (2009), 1550–1553.

[22] J. Hipp, U. Güntzer, and G. Nakhaeizadeh. 2000. Algorithms for association rule mining—a general survey and comparison. *ACM SIGKDD Explorations Newsletter* 2, 1 (2000), 58–64.

[23] Victoria Hodge and Jim Austin. 2004. A survey of outlier detection methodologies. *Artificial intelligence review* 22, 2 (2004), 85–126.

[24] Jiansheng Huang, Ting Chen, AnHai Doan, and Jeffrey F. Naughton. 2008. On the provenance of non-answers to queries over extracted data. *PVLDB* 1, 1 (2008), 736–747.

[25] Manas Joglekar, Hector Garcia-Molina, and Aditya G. Parameswaran. 2016. Interactive data exploration with smart drill-down. In *ICDE*. 906–917.

[26] Bhargav Kanagal, Jian Li, and Amol Deshpande. 2011. Sensitivity analysis and explanations for robust query evaluation in probabilistic databases. In *SIGMOD*. 841–852.

[27] Nodira Khoussainova, Magdalena Balazinska, and Dan Suciu. 2012. PerfXplain: Debugging MapReduce Job Performance. *PVLDB* 5, 7 (2012), 598–609.

[28] Seokki Lee, Sven Köhler, Bertram Ludäscher, and Boris Glavic. 2017. A SQL-Middleware Unifying Why and Why-Not Provenance for First-Order Queries. In *ICDE*. 485–496.

[29] Seokki Lee, Xing Niu, Bertram Ludäscher, and Boris Glavic. 2017. Integrating Approximate Summarization with Provenance Capture. In *TaPP*.

[30] Alexandra Meliou, Wolfgang Gatterbauer, Katherine F. Moore, and Dan Suciu. 2010. The Complexity of Causality and Responsibility for Query Answers and non-Answers. *PVLDB* 4, 1 (2010), 34–45.

[31] Alexandra Meliou and Dan Suciu. 2012. Tiresias: the database oracle for how-to queries. In *SIGMOD*. 337–348.

[32] Douglas C Montgomery, Elizabeth A Peck, and G Geoffrey Vining. 2012. *Introduction to linear regression analysis*. Vol. 821. John Wiley & Sons.

[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. *PVLDB* 8, 10 (2015), 1082–1093.

[34] Karl Pearson. 1900. X. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 50, 302 (1900), 157–175.

[35] Sudeepa Roy, Laurel Orr, and Dan Suciu. 2015. Explaining query answers with explanation-ready databases. *PVLDB* 9, 4 (2015), 348–359.

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

[37] Sunita Sarawagi. 1999. Explaining Differences in Multidimensional Aggregates. In *VLDB*. 42–53.

[38] Sunita Sarawagi, Rakesh Agrawal, and Nimrod Megiddo. 1998. Discovery-Driven Exploration of OLAP Data Cubes. In *EDBT*. 168–182.

[39] Sunita Sarawagi and Gayatri Sathe. 2000. i^3: Intelligent, Interactive Investigaton of OLAP data cubes. In *SIGMOD*. 589.

[40] Gayatri Sathe and Sunita Sarawagi. 2001. Intelligent Rollups in Multidimensional OLAP Data. In *VLDB*. 531–540.

[41] Tarique Siddiqui, Albert Kim, John Lee, Karrie Karahalios, and Aditya G. Parameswaran. 2016. Effortless Data Exploration with zenvisage: An Expressive and Interactive Visual Analytics System. *PVLDB* 10, 4 (2016), 457–468.

[42] Chris Stolte, Diane Tang, and Pat Hanrahan. 2002. Polaris: A System for Query, Analysis, and Visualization of Multidimensional Relational Databases. *IEEE Trans. Vis. Comput. Graph.* 8, 1 (2002), 52–65.

[43] Balder ten Cate, Cristina Civili, Evgeny Sherkhonov, and Wang-Chiew Tan. 2015. High-Level Why-Not Explanations Using Ontologies. In *PODS*. 31–43.

[44] Quoc Trung Tran and Chee-Yong Chan. 2010. How to ConQueR Why-not Questions. In *SIGMOD*. 15–26.

[45] Moshe Y. Vardi. 1982. The Complexity of Relational Query Languages (Extended Abstract). In *STOC*. 137–146.

[46] Kanit Wongsuphasawat, Zening Qu, Dominik Moritz, Riley Chang, Felix Ouk, Anushka Anand, Jock D. Mackinlay, Bill Howe, and Jeffrey Heer. 2017. Voyager 2: Augmenting Visual Analysis with Partial View Specifications. In *CHI*. 2648–2659.

[47] Eugene Wu and Samuel Madden. 2013. Scorpion: Explaining Away Outliers in Aggregate Queries. *PVLDB* 6, 8 (2013), 553–564.

	Explanation				
Rank	author	venue	year	count(*)	score
1	A_X		2013	43	39.6
2	A_X	TKDE	2012	1	5.8
3	A_X	ICDE	2013	1	4.8
4	A_X	SIGMOD	2012	1	4.5
5	A_X	SIGMOD	2013	1	3.3

Table 4: Top-5 explanations produced by CAPE for $\phi_0 = (Q_0, (A_X, SIGKDD, 2012, 6), high)$

	Explanation				
rank	type	community	year	count(*)	score
1		26	2012	117	63.9
2	Battery	26	2012	24	63.0
3	Battery	25	2011	79	60.5
4	Battery		2010	1095	49.0
5	Assault	26	2011	10	40.1

Table 5: Top-5 explanations produced by CAPE for $\phi_1 = (Q_{Crime}, (Battery, 26, 2011, 16), low)$

	Explanation				
Rank	author	venue	year	count(*)	score
1	A_X	WSDM	2012	1	44.7
2	A_X	TIST	2012	1	44.7
3	A_X	SYNTHESIS	2012	1	44.7
4	A_X	RTSS	2012	1	44.7
5	A_X	PIEEE	2012	1	44.7

Table 6: Top-5 explanations produced by the baseline method for $\phi_0 = (Q_0, (A_X, SIGKDD, 2012, 6), high)$

	Explanation				
rank	type	community	year	count(*)	score
1	Battery	25	2011	79	62.2
2	Narcotics	25	2011	94	54.9
3	Battery	25	2010	74	46.9
4	Narcotics	25	2010	87	44.8
5	Battery	25	2012	66	40.4

Table 7: Top-5 explanations produced by the baseline method for $\phi_1 = (Q_{Crime}, (Battery, 26, 2011, 16), low)$

A QUALITATIVE EVALUATION

A.1 Example Explanations

We now discuss explanations for the DBLP and Crime datasets. A brief qualitative evaluation was already given in the introduction. Here we discuss explanations returned by CAPE for two additional user questions: 1) over the running example query Q_0 which computes the number of publications per author, venue, and year and 2) over a query for the crime dataset which computes the number of crimes per crime type, community area and year.

DBLP dataset. For the DBLP dataset, the top-5 explanations for $(Q_0(Pub), (A_X, SIGKDD, 2012, 6), high)$, i.e., *"Why is the number of A_X's SIGKDD 2012 papers high?"*, are shown in Table 4. The explanation ranked 1^{st} explains this outcome by the low number of papers A_X published in 2013 which may imply that A_X focused on publishing in 2012 which in turn explains the large amount of SIGKDD papers in 2012. The remaining 4 explanations point out lower than expected number of publications in a specific venue in 2012 or 2013.

Crime dataset. The top-5 explanations for the user question $(Q(Crime), (Battery, 26, 2011, 16), low)$, i.e., *"Why is the number of crimes of type Battery in community area 26 in 2011 low?"*, are shown in Table 5. The highest ranked explanation explains this outcome by the high number of crimes in this area in 2012. Explanation 2 is a more specific version of the first explanation implying that low numbers in 2011 are related to the high number of batteries in this area in 2012. Explanation 3 reasons that the low number of batteries in area 26 in 2011 can be explained by a high number of batteries in area 25 — one of the 4 adjacent areas of area 26 — in 2011, while explanation 5 relates it to the high number of assaults in that area in 2011. Explanation 4 indicates that the high number of batteries in 2010 may have led to the low number in 2011 due to, e.g., increased police presence.

A.2 Comparison with a Baseline Approach

We also compare the explanations from our method with the explanations from a baseline method, which finds counterbalances in the query result using deviation (from the average value in the query result) and similarity to score explanations. Explanations produced by the baseline method are shown in Table 6 and 7. Note that this method prefers tuples from the query result as explanations if their absolute values are high/low. Consider the explanations for $\phi_1 = (Q_{Crime}, (Battery, 26, 2011, 16), low)$ shown in Table 7. All explanations are in the community area 25, because it is adjacent to area 26 and has the largest number of crimes. By not considering patterns the baseline method is ignorant to whether a high/low value is an outlier (differs from an expected value) or not. For instance, narcotics violations and batteries in area 25 are in general high, but not higher than usual for most of the returned explanations. This effect is also observed in the explanations for the DBLP user question produced by the baseline method. Here the explanations only cover venues that A_X rarely publishes in (low count, but predictably so). However, these are unlikely to have caused the outcome of interest. Finally, the baseline method does not consider other aggregation queries with different group-by attributes and, thus, will miss explanations that are more coarse-grained or more finer-grained than the results returned by the query in the user question (e.g., the high total number of batteries in 2010).

B USER STUDY

We conducted a user study to evaluate the utility of CAPE. Specifically, we want to investigate whether CAPE (S1) provides meaningful explanations, and (S2) whether CAPE reduces the amount of time a user needs to find and confirm explanations.

Algorithm 3 Brute Force Pattern Discovery

```
NaivePatternDiscovery($R, \theta, \delta, \lambda, \Delta$)
1  $\mathcal{P} = \emptyset$
2  for $M \in \mathcal{M}, agg \in \mathcal{A}, A \in (R \cup \{*\})$
3      for $G \subset (R - a)$
4          for $F \subset G$
5              $V = G - F$
6              $P = (F, V, agg, A, M)$
7              if NaivePatternHolds($P, R, \theta, \delta, \lambda, \Delta$)
8                  $\mathcal{P} = \mathcal{P} \cup \{P\}$
9  return $\mathcal{P}$
```

Algorithm 4 Evaluating Whether a Pattern Holds Globally

```
NaivePatternHolds($P, R, \theta, \delta, \lambda, \Delta$)
1  $frag_{good} = \emptyset, frag_{supp} = \emptyset$
2  for $f \in \pi_F(R)$
3      $Q_{P,f} = \gamma_{V,agg(A)}(\sigma_{F=f}(R))$
4      for $t \in Q_{P,f}(R)$
5          $h_{P,f}(t[V]) = t[agg(A)]$
6      if $|Q_{P,f}(R)| \geq \delta$
7          $frag_{supp} = frag_{supp} \cup \{f\}$
8          $GoF$ = ApplyRegression($M, h_{P,f}$)
9          if $GoF > \theta$
10             $frag_{good} = frag_{good} \cup \{f\}$
11 if $|frag_{good}| \geq \Delta \wedge \frac{|frag_{good}|}{|frag_{supp}|} \geq \lambda$
12     return TRUE
13 else return FALSE
```

Algorithm 5 Explore sort orders for a set of group-by of G

```
ExploreSortOrders($G, D, C, \mathcal{P}, R, \theta, \delta, \lambda, \Delta$)
1  for each permutation $S$ of $G$
2      if $\exists (F, V) \notin C : F \cup V = G \wedge$ isPrefix($F, S$)
3          $Q_{sort}$ = SELECT * FROM D ORDER BY S
4          $D_{sort} = Q_{sort}(D)$
5          for each $F, V$ where $F \cup V = G \wedge$ isPrefix($F, S$)
6              $C = C \cup \{(F, V)\}$
7              if $(F, V) \notin C \wedge$ ValidateFDs($F, V, \Psi$)
8                  for $M \in \mathcal{M}, (agg \in \mathcal{A}, a \in (R))$
                           $\vee (agg = count, a = *)$
9                      $P = (F, V, agg, A, M)$
10                     FitPattern($P, D_{sort}, \theta, \delta, \lambda, \Delta$)
11 return $(\mathcal{P}, C)$
```

Datasets and queries. We use a subset of the Chicago Crime dataset that contains only 2 community areas, and an aggregation query Q that computes the number of crimes per primary type, location, and year.

```
Q = SELECT primary_type, location_desc, year, count(*)
    FROM crime
    GROUP BY primary_type, location_desc, year
```

Participants. There are 14 participants - all of them are graduate students working on different topics in databases, and thus they all have some prior experience with SQL and are capable to finish the tasks in our user study. We assigned our participants randomly to either the "treatment" group or the "control" group.

Tasks. Each study participant is asked to find explanations to the following three questions based on the result of query Q over Crime table C.

$\phi_1 = (Q, C, t_1, high), t_1 =$ `(Assault,CTA bus,2008,5)`

$\phi_2 = (Q, C, t_2, low), t_2 =$ `(Battery,Garage,2009,11)`

$\phi_3 = (Q, C, t_3, high), t_3 =$ `(Crim. Damage,Church,2006,12)`

Both groups were allowed to explore the dataset using SQL queries, but only the "treatment" group was additionally provided with the top-10 explanations produced by Cape for each question. Before the start of a session, we allowed participants to familiarize themselves with the dataset as well as with the user questions. Then, we gave each participant 35 minutes to find explanations. The treatment group was instructed that the explanations provided by Cape were not necessarily correct and that it may be important to confirm explanations by running queries on their own.

After finishing the task, the control group was provided with the explanations produced by Cape, and each participant was asked to rate on a scale of 1 (strongly disagree) to 10 (strongly agree) how helpful these explanations were/would have been for (S1) finding a meaningful explanation, and for (S2) finding explanations faster. We also asked participants to provide additional comments/suggestions.

Results. Overall, the responses were very positive: 8.86 ± 1.87 for S1 and and 8.36 ± 1.39 for S2 (mean $\pm$ standard deviation). To evaluate the performance of the participants, we manually determined a set of sensible explanations (not necessarily returned by Cape) for each question. A question is considered to be correctly answered if any explanation found by the participant was in this set. The success rates of the two groups are shown below. Question ϕ_3 is a less extreme outlier and, thus, the overall success rate was lower (additionally, some participants ran out of time). In short, having access to the explanations produced by Cape improved the performance of participants, further demonstrating the quality of the explanations produced by Cape.

	Success rate		
Group	ϕ_1	ϕ_2	ϕ_3
Treatment	86%	71%	57%
Control	71%	43%	0%

C PSEUDOCODE OF THE ALGORITHMS

Pseudocode for additional algorithms from Section 3 and 4.

D FUNCTIONAL DEPENDENCIES

Functional dependencies (FDs) that hold for an input table can be used to infer that a pattern holds if another pattern

Algorithm 6 Checking whether pattern holds (Data given)

```
FitPattern($P$, $D_{sort}$, $\theta$, $\delta$, $\lambda$, $\Delta$)
1  $frag_{good} = \emptyset, frag_{supp} = \emptyset, f = $ NIL$, D_f = \emptyset, h_f = \emptyset$
2  for $t \in D_{sort}$
3      $f_{cur} = t[F]$
4      if $f_{cur} = f$          // Collect data for $f$
5          $h_f(t[V]) = t[agg(A)]$
6          $D_f = D_f \cup \{t\}$
7      elseif $f \neq$ NIL       // Regression and check GoF
8          if $|D_f| \geq \delta$
9              $frag_{supp} = frag_{supp} \cup \{f\}$
10             $GoF = $ ApplyRegression$(M, h_f)$
11             if $GoF > \theta$
12                 $frag_{good} = frag_{good} \cup \{f\}$
13 if $|frag_{good}| \geq \Delta \wedge \frac{|frag_{good}|}{|frag_{supp}|} \geq \lambda$
14     return TRUE
15 else return FALSE
```

is already known to hold. In the following, we first prove an inference rule that augments the partition attributes of a pattern based on a set of FDs and then motivate why inferred patterns are redundant in terms of explanatory power and, thus, can be safely excluded from pattern detection without affecting our ability to generate explanations.

Augment F. Consider a functional dependency $A \longrightarrow B$ where $A \subset R$ and $B \in R$ for a relation R. Furthermore, consider a pattern $P = [F] : V \rightsquigarrow agg(A)$ that holds globally over R where $A \subseteq F$. If we add B to F to get a pattern $P' = [F'] : V \rightsquigarrow agg(A)$ where $F' = F \cup \{B\}$, then P' is guaranteed to hold globally. We can state this as the inference rule shown as Equation 2 below.

$$\begin{aligned} &R \models_{(\theta,\delta),(\lambda,\Delta)} [F] : V \rightsquigarrow agg(A) \wedge A \subseteq F \wedge A \longrightarrow B \\ \Leftrightarrow &R \models_{(\theta,\delta),(\lambda,\Delta)} [F'] : V \rightsquigarrow agg(A) \end{aligned} \tag{2}$$

To see why this rule holds, consider an $f \in frag(P, R)$ and the queries $Q_{P,f} := \gamma_{\mathsf{V},\mathsf{agg(A)}}(\sigma_{F=f}(R))$. Since, $A \rightarrow B$ and $A \subseteq F$, we know that for every tuple in R where $t[F] = f$ we also have $t[B] = b$ for some value b. Now since $F = F \cup \{B\}$, if we set $f' = (f, b)$, then the set of tuples in R where $t[F'] = f'$ is the same as the set of tuples where $t[F] = f$. It follows that queries $Q_{P,f} := \gamma_{\mathsf{V},\mathsf{agg(A)}}(\sigma_{F=f}(R))$ and $Q_{P',f} := \gamma_{\mathsf{V},\mathsf{agg(A)}}(\sigma_{F'=f'}(R))$ produce the same result. Thus, the pattern P holds locally on f if and only if P' holds locally on f', because the regression analysis that is used to determine whether a pattern holds locally is performed on the results of these queries.

Skipping redundant patterns. An important property of the augmentation rule shown in Equation 2 is that if an explanation for a user question is based on an augmented pattern P' then the same type of explanations hold for the pattern P that was augmented. The opposite is not necessarily true, because the augmented pattern may contain attributes that are not part of the group-by of the user question and we only consider patterns for explanations whose G_P are contained in the set of group-by attributes of the user question. Therefore, it would be sufficient to evaluate pattern candidates where the set of partition attributes F is minimized w.r.t. a set of functional dependencies that hold over the input dataset. To this end, we use functional dependencies to decide if we want to do regression on a certain pattern based on whether F is minimal, i.e., none of the attributes in F is implied by a subset of F which can be checked in linear time in the number of attributes by testing for each $A \in F$ whether $A \in (F - \{A\})^+$ (i.e., A is in the attribute closure of $F - \{A\}$). Given a pattern and set of FDs Ψ as input, for every $A \in F$ we check whether the other attributes $F - \{A\}$ imply A. If we find an implied attribute then we return FALSE (the pattern should not be considered).[4] If no such attribute exists, then we consider the pattern (return TRUE).

F functionally determines V. Consider a pattern P where F functionally determines V ($F \longrightarrow V$). Then for every $f \in frag(R, P)$, the fragment of R for f will contain only one row since F is a key for the fragment. That is, such a pattern cannot possibly fulfill the local support threshold δ as long as $\delta > 1$ (which is reasonable to assume).

Detecting functional dependencies during pattern mining. Functional dependencies are often not readily available for a dataset. Obviously, we could employ any existing FD detection algorithm to determine the set of FDs for the input relation R (e.g, see [33] for a performance comparison of state-of-the-art approaches). Alternatively, we can compute functional dependencies as a side effect of pattern mining. When we evaluate an aggregation query with group-by attributes G to gather the data for regression analysis for patterns with $G_P = (F \cup V) = G$, then we can count the groups to determine the number of unique G values in the dataset. We memorize this information to test an FD $A \longrightarrow B$ where $A \subseteq R$ and $B \subseteq R$ holds. We check whether $|\pi_A(R)| = |\pi_{A \cup B}(R)|$ using the number of groups for A and $A \cup B$. Recall that $A \longrightarrow A_1$ and $A \longrightarrow A_2$ implies $A \longrightarrow A_1, A_2$ and vice versa (follows from Armstrong's axioms). Thus, it is sufficient to check for FDs with a single RHS attribute. If we detect FDs during pattern mining to avoid the cost of mining FDs, then we have to ensure that we have evaluated an FD $A \longrightarrow B$ before it is needed for pattern skipping. This can be achieved by mining patterns in increasing size of their group-by attributes G_P.

[4]This can be implemented using the concept of the closure of a set of attributes H with respect to a set of FDs Ψ, which is the set of attributes that are implied by H according to Ψ.