A Plaque Test for Redundancies in Relational Data

Christoph Köhnen[1], Stefan Klessinger[1], Jens Zumbrägel[1] and Stefanie Scherzinger[1]

[1]University of Passau, Passau, Germany

Abstract

Inspired by the visualization of dental plaque at the dentist's office, this article proposes a novel visualization of redundancies in relational data. Our approach is based on a well-principled information-theoretic framework that has seen limited practical application in systems and tools. In this framework, we quantify the information content (or entropy) of each cell in a relation instance, given a set of functional dependencies. The entropy value signifies the likelihood of recovering the cell value based on the dependencies and the remaining tuples. By highlighting cells with lower entropy, we effectively visualize redundancies in the data. We present an initial prototype implementation and demonstrate that a straightforward approach is insufficient to handle practical problem sizes. To address this limitation, we propose several optimizations, which we prove to be correct. Additionally, we present a Monte Carlo approximation with a known error, enabling computationally tractable analysis. By applying our visualization technique to real-world datasets, we showcase its potential. Our vision is to empower data analysts by directing their focus in data profiling toward pertinent redundancies, analogous to the diagnostic role of a plaque test at the dentist's office.

1. Introduction

Database normalization is a well-studied field, with the theory of functional dependencies as a cornerstone, cf. [1]. There are several proposals to visualize functional dependencies, such as sunburst diagrams or graph-based visualizations [2]. Yet these approaches only visualize the dependencies, irrespective of the data instance.

We propose a novel visualization that reveals the redundancies captured by functional dependencies. We refer to our approach as "plaque test": Like a plaque test at the dentist's which colors dental plaque to reveal unwanted residue on patients' teeth, our plaque test reveals redundancies in relational data. The plaque test may be applied in data profiling, when dependencies have been discovered, before carrying out data cleaning tasks, or when preparing towards schema normalization — in all these scenarios, domain experts will want to explore the redundancies resident in their data.

Formally, our approach leverages an existing information-theoretic framework by Arenas and Libkin [3]. To our knowledge, we are the first to implement and apply this framework to a practical problem.

Next, we illustrate our concept of plaque tests.

Example 1.1. Figure 1a shows an example from the German Wikipedia site on database normalization[1]. The relation manages a CD collection. Each CD has an identifier (ID), an AlbumTitle, and was released in a given year (RYear). The band who released the CD (Band) was also founded in a given year (BYear). Each CD has several tracks (Track) and each track has a title (TrackTitle).

We consider the following functional dependencies[2]:

$$\text{ID} \rightarrow \text{AlbumTitle, Band, BYear, RYear}$$

$$\text{ID, Track} \rightarrow \text{TrackTitle} \quad \text{and} \quad \text{Band} \rightarrow \text{BYear}$$

Figure 1b shows the plaque test applied to the instance from Figure 1a, based on these dependencies: Each cell is assigned an entropy value. Our plaque test then visualizes these entropy values by coloring the cells: the more redundant the value, the smaller the entropy value and the deeper the blue.

Specifically, let us consider the dependency "ID → AlbumTitle". In the given data instance, the title "Not That Kind" of the album with ID 1 is recorded redundantly. If the title were lost in the first tuple, it could still be recovered from the second or third tuple (and vice versa). This is captured by an entropy of 0.8 and blue color (denoting "plaque"). On the contrary, if the title "Freak of Nature" of the album with ID 3 were lost, it could not be recovered. Consequently, the cell has entropy 1 and remains white ("no plaque").

Let us next consider the dependency "Band → BYear", stating that the band determines the year of foundation. Since there are four tuples for the band Anastacia, the year of the foundation is more redundant than the year of the release of her album "Not That Kind" (recorded in three tuples). Accordingly, the entropy values differ and our plaque test colors the foundation years for Anastacia's band in a deeper blue.

Finally, let us consider the dependency "ID → BYear". Since we have three entries for the album with ID 1 ("Not

Joint Workshops at 49th International Conference on Very Large Data Bases (VLDBW'23) — the 12th International Workshop on Quality in Databases (QDB'23), August 28 - September 1, 2023, Vancouver, Canada

christoph.koehnen@uni-passau.de (C. Köhnen);
stefan.klessinger@uni-passau.de (S. Klessinger);
jens.zumbraegel@uni-passau.de (J. Zumbrägel);
stefanie.scherzinger@uni-passau.de (S. Scherzinger)

[1]Retrieved from https://de.wikipedia.org/wiki/Normalisierung_(Datenbank), last accessed July 2023.

[2]Note that this is not a canonical cover of the dependencies.

ID	AlbumTitle	Band	BYear	RYear	Track	TrackTitle
1	Not That Kind	Anastacia	1999	2000	1	Not Th…
1	Not That Kind	Anastacia	1999	2000	2	I'm Out…
1	Not That Kind	Anastacia	1999	2000	3	Cowboy…
2	Wish You Were Here	Pink Floyd	1965	1975	1	Shine O…
3	Freak of Nature	Anastacia	1999	2001	1	Paid my…

(a) The relational input data.

ID	AlbumTitle	Band	BYear	RYear	Track	TrackTitle
1	0.8	0.8	0.6	0.8	1	1
1	0.8	0.8	0.6	0.8	1	1
1	0.8	0.8	0.6	0.8	1	1
1	1	1	1	1	1	1
1	1	1	0.7	1	1	1

(b) Entropies for 6 unary FDs.

ID	AlbumTitle	Band	BYear	RYear	Track	TrackTitle
0.6	0.6	0.4	0.4	0.6	1	1
0.6	0.6	0.4	0.4	0.6	1	1
0.6	0.6	0.4	0.4	0.6	1	1
1	1	1	1	1	1	1
1	1	0.7	0.7	1	1	1

(c) Entropies for 23 unary FDs.

Figure 1: Plaque tests for the original relation (top) with genuine functional dependencies (middle) and automatically discovered functional dependencies (bottom). Cell color/hue corresponds to entropy values.

That Kind"), and only one entry each for the albums with ID 2 ("Wish You Were Here") and ID 3 ("Freak of Nature"), the band year is even more redundant for Anstacia's album "Not That Kind" (entropy value 0.6) than for her album "Freak of Nature" (entropy value 0.7).

So far, we only considered genuine dependencies. Next, we assume that the dependencies have been automatically discovered during data profiling.

Example 1.2. The Metanome tool [4] discovers 23 dependencies in this relation instance. This includes the cyclic dependencies "Band → BYear" and "BYear → Band".

Figure 1c shows the result of our plaque test, given this new set of functional dependencies. Throughout, the coloring of the cells is darker. Moreover, more cells are colored. This reveals that the instance now contains additional redundancies.

Notably, the entropy value for the band name "Anastacia" is as low as 0.4, since it may be recovered from several dependencies, such as "ID → Band" and "BYear → Band". Thus, plaque is additive when an attribute occurs on the right-hand side of several functional dependencies.

As these examples illustrate, the plaque test visualizes redundancies in a well-principled fashion. It is highly sensitive to the set of functional dependencies and the values that appear in the data instance.

Contributions

- We apply an existing information-theoretical framework for a "plaque test" that visualizes redundancies inherent in relational data.
- We show that a straightforward implementation for computing the entropy values underlying our "plaque test" does not scale beyond toy examples.
- Thus we propose several effective optimizations:
  1. We discuss scenarios where entropy values can be immediately assigned to 1, thereby skipping computation, and where one can focus on a subset of the data instance when computing entropy values. We prove the correctness of these optimizations.
  2. We present a Monte Carlo approximation to compute entropy values. We provide the formula to determine the number of iterations required to achieve a given accuracy with a certain confidence.
- We present visual plaque tests for several real-world datasets and discuss how the discovered "plaque" can indeed be helpful for data exploration.
- We conduct run-time experiments with our implementation and explore the effect of our optimizations.

Reproducibility Our research artifacts, including our prototype implementation, can be found at https://doi.org/10.5281/zenodo.8220684.

Long version Further experiments and the full set of proofs can be found in the long version of this article at https://doi.org/10.48550/arXiv.2306.02890.

Structure We provide preliminaries on functional dependencies, entropy, and information content in Section 2. In Section 3 we introduce two lines of optimization, one is an exact method, and the other an approximation. We present our experiments in Section 4. We discuss related work in Section 5 and conclude in Section 6.

2. Preliminaries

When introducing functional dependencies in Section 2.1, we deviate from the common definition (cf. [1]) in that we take into account the order of tuples. This allows us to identify individual cells in a relation instance. The notion of entropy-related information content originates from the work by Arenas and Libkin [3] (Section 2.2). As a first contribution, we present simplifications to compute the corresponding entropy values (Section 2.3).

2.1. Functional Dependencies

Denote by $\mathbb{N}$ the set of positive integers, $\mathbb{N} := \{1, 2, \dots\}$.

Definition 2.1. A *relation* R of arity m is specified by a finite set $\mathrm{sort}(R) := \{A_1, \dots, A_m\}$ of attributes A_i. The domain of an attribute A is denoted by $\mathrm{Dom}(A)$.

An *instance* I of a relation R is a partial map

$$I \colon \mathbb{N} \rightharpoonup \bigtimes_{A \in \mathrm{sort}(R)} \mathrm{Dom}(A)$$

with finite domain of definition $\mathrm{Def}(I) := \{j \in \mathbb{N} \mid I(j) \text{ is defined}\}$.

The above definition of a relation instance allows for duplicate tuples and preserves the order of the tuples.

To simplify the exposition, in this work we assume that $\mathrm{Dom}(A) = \mathbb{N}$ for all $A \in \mathrm{sort}(R)$, i.e., each tuple consists of positive integers. Thus an instance I of a relation R of arity m can be seen as just a partial map $I \colon \mathbb{N} \rightharpoonup \mathbb{N}^m$.

Definition 2.2. Let R be a relation and I an instance of R. For a subset $K := \{A_1, \ldots, A_s\} \subseteq \mathrm{sort}(R)$ of attributes, we have the projection $\pi_K(I)$ of the tuples in I to the attributes in K. Moreover, for a subset $J \subseteq \mathrm{Def}(I)$ we have the restricted map $I|_J \colon J \to \mathbb{N}^m$ defined for all indices $j \in J$.

Given an instance I of a relation R we denote by $t_j[A_k]$ the value of the attribute A_k in the j-th tuple $t_j := I(j)$ of I.

Similarly, for a subset $K = \{A_1, \ldots, A_s\} \subseteq \mathrm{sort}(R)$ of attributes we denote by $t_j[A_1 \ldots A_s]$ the tuple of values $t_j[A_1], \ldots, t_j[A_s]$, that is, the j-th tuple $\pi_K(I(j))$ of the projection $\pi_K(I)$.

Definition 2.3. A *functional dependency* in a relation R is a pair $f := (\{A_1, \ldots, A_s\}, B)$, where $A_1, \ldots, A_s, B \in \mathrm{sort}(R)$ are attributes. We write such a pair as $A_1 \ldots A_s \to B$.

An instance I of a relation R is said to *fulfill* the functional dependency f (write $I \models f$) if for all $j_1, j_2 \in \mathrm{Def}(I)$ it holds

$$t_{j_1}[A_1 \ldots A_s] = t_{j_2}[A_1 \ldots A_s] \;\Rightarrow\; t_{j_1}[B] = t_{j_2}[B].$$

Moreover, the instance I *fulfills* a set F of functional dependencies (write $I \models F$) if $I \models f$ for all $f \in F$.

We also consider relation instances having unspecified values at some positions. Let $\mathbf{var}$ be a (countably infinite) set of variables.

Definition 2.4. Let I be an instance of a relation R with attributes $\mathrm{sort}(R) = \{A_1, \ldots, A_m\}$. A *position* in I is a pair (j, A_k) with $j \in \mathrm{Def}(I)$ and $k \in \{1, \ldots, m\}$; it represents the cell of the k-th attribute of the j-th tuple, with value $t_j[A_k]$. The instance obtained by putting the value v at position $p = (j, A_k)$ is denoted by $I_{p \leftarrow v}$.

Let $Q = \{q_1, \ldots, q_k\}$ be a set of positions, $X = (x_1, \ldots, x_k)$ distinct variables, and $V = (v_1, \ldots, v_k)$ values in $\mathbb{N}$. The instance obtained by replacing each position q_i by the variable x_i resp. by value v_i for $1 \leqslant i \leqslant k$ is denoted by $I_{Q \leftarrow X}$ resp. $I_{Q \leftarrow V}$ (hence, the instance obtained by replacing the positions from Q by variables, and then position p by the value v, is $(I_{Q \leftarrow X})_{p \leftarrow v}$).

An instance I containing distinct variables at positions $Q = \{q_1, \ldots, q_k\}$ *fulfills* a set F of functional dependencies (write $I \models F$) if there exists a set of values $V = (v_1, \ldots, v_k)$ such that $I_{Q \leftarrow V} \models F$. For one functional dependency $f \in F$ we write $I \models f$ if $I \models \{f\}$.

For $f := A_1 \ldots A_s \to B$ (and I possibly containing variables) we immediately obtain that $I \models f$ holds if and only if for all $j_1, j_2 \in \mathrm{Def}(I)$ such that $t_{j_1}[B] \notin \mathbf{var}$ and $t_{j_2}[B] \notin \mathbf{var}$ there holds

$$t_{j_1}[A_1 \ldots A_s] = t_{j_2}[A_1 \ldots A_s] \;\Rightarrow\; t_{j_1}[B] = t_{j_2}[B].$$

So the above definition requires a single functional dependency to be fulfilled only for tuples without variables. Indeed, since all variables are distinct, it is always possible to set values in their positions so that the functional dependency is fulfilled.

For a set F of functional dependencies, it can be shown that $I \models F$ if and only if $I \models f$ for all $f \in F^*$, where F^* is the transitive closure of F. This equivalence ensures the same semantics as in the original work by Arenas and Libkin [3] and we assume that the transitive closure of functional dependencies is provided.

2.2. Entropy and Information Content

In the following, we deal with discrete probability spaces $\mathcal{X} = (X, P_{\mathcal{X}})$ on finite sets X, where $P_{\mathcal{X}}(x)$ for $x \in X$ denotes the probability of the event $\{x\}$. The information-theoretic *entropy* of the probability space $\mathcal{X}$ is given by

$$H(\mathcal{X}) := -\sum_{x \in X} P_{\mathcal{X}}(x) \log P_{\mathcal{X}}(x).$$

If $\mathcal{A} = (A, P_{\mathcal{A}})$ and $\mathcal{B} = (B, P_{\mathcal{B}})$ are probability spaces on finite sets A and B with joint distribution $P_{\mathcal{A} \times \mathcal{B}}$, then the *conditional probability* of $a \in A$ given $b \in B$ is defined by

$$P(a|b) := \frac{P_{\mathcal{A} \times \mathcal{B}}(a, b)}{P_{\mathcal{B}}(b)}$$

provided that $P_{\mathcal{B}}(b)$ is non-zero.

Conversely, the conditional probabilities $P(a|b)$ with the probabilities $P_{\mathcal{B}}(b)$ determine the joint distribution $P_{\mathcal{A} \times \mathcal{B}}$ by the above formula.

Definition 2.5. Let $\mathcal{A} = (A, P_{\mathcal{A}}), \mathcal{B} = (B, P_{\mathcal{B}})$ be probability spaces. The *conditional entropy* of $\mathcal{A}$ given $\mathcal{B}$ is

$$H(\mathcal{A} \mid \mathcal{B}) := -\sum_{b \in B} P_{\mathcal{B}}(b) \sum_{a \in A} P(a|b) \log P(a|b).$$

This value describes the remaining uncertainty in probability space $\mathcal{A}$ given the outcome in $\mathcal{B}$. If the probability distributions of $\mathcal{A}$ and $\mathcal{B}$ are independent, then $P(a|b) = P_{\mathcal{A}}(a)$ for all $b \in B$, and thus we obtain $H(\mathcal{A} \mid \mathcal{B}) = -\sum_{a \in A} P_{\mathcal{A}}(a) \log P_{\mathcal{A}}(a) = H(\mathcal{A})$.

Consider now a relation instance I of arity m and denote by $\mathrm{Pos} := \mathrm{Def}(I) \times \{A_1, \ldots, A_m\}$ its set of positions. Let F be a set of functional dependencies fulfilled by I and $p \in \mathrm{Pos}$ a position. We define two probability spaces as follows.

First, let $\mathcal{B}(I,p) := (\mathcal{P}(\mathrm{Pos}\setminus\{p\}), P_{\mathcal{B}})$, where $P_{\mathcal{B}}$ is the uniform distribution on the set of all subsets of $\mathrm{Pos}\setminus\{p\}$. This space models the possible cases when we lose a set of possible values from the instance I on positions other than the considered one p.

Then, for $k \in \mathbb{N}$ we let $\mathcal{A}^k_F(I,p) := (\{1,\ldots,k\}, P_{\mathcal{A}})$, where the conditional probability of $v \in \{1,\ldots,k\}$ given $Q \subseteq \mathrm{Pos}\setminus\{p\}$ is

$$P(v|Q) := \begin{cases} 1/\#V_Q & \text{if } v \in V_Q, \\ 0 & \text{otherwise,} \end{cases}$$

with $V_Q := \{v \in \{1,\ldots,k\} \mid (I_{Q\leftarrow X})_{p\leftarrow v} \models F\}$. This probability space models the possible values in $\{1,\ldots,k\}$ to be put in at position p for which we lost the value, when Q is the set of positions of the other lost values in the instance.

Definition 2.6. Let F be a set of functional dependencies for a relation R and let I be an instance of R with $I \models F$. The *information content* of position p with respect to F in instance I is given as

$$\mathrm{INF}_I(p \mid F) := \lim_{k\to\infty} \frac{\mathrm{INF}^k_I(p \mid F)}{\log k},$$

where $\mathrm{INF}^k_I(p \mid F) := H(\mathcal{A}^k_F(I,p) \mid \mathcal{B}(I,p))$ is the conditional entropy of the probability space modeling the possible values for the considered position p given the space modeling the possible sets of other lost values in the instance.

Unfortunately, when using the above formula for the conditional entropy $\mathrm{INF}^k_I(p \mid F)$ directly, the computation grows exponentially with the number of cells in the given instance. Indeed, for each cell except the one at position p the value can be deleted or not, so every subset of $\mathrm{Pos}\setminus\{p\}$ is an elementary event in the probability space $\mathcal{B}(I,p)$. Therefore, each additional cell in the instance I doubles the number of events to be taken into account for the computation of the information content.

2.3. Simplifications

We now provide more compact and simplified (but still exponentially complex) formulas for the information content. The first result easily follows from the definition of conditional entropy.

Proposition 2.7. Let F be a set of functional dependencies and I an instance with $I \models F$. Then the information content of a position p in I with respect to F is given by

$$\mathrm{INF}_I(p \mid F) = \frac{1}{2^{\#\,\mathrm{Pos}\,-1}} \sum_{Q\subseteq \mathrm{Pos}\setminus\{p\}} \lim_{k\to\infty} \frac{\log \#V_Q}{\log k},$$

where $V_Q := \{v \in \{1,\ldots,k\} \mid (I_{Q\leftarrow X})_{p\leftarrow v} \models F\}$.

The values $\mathrm{INF}_I(p \mid F, Q) := \lim_{k\to\infty} \frac{\log \#V_Q}{\log k}$ can be seen as the information content of p in I with respect to F given a fixed subset $Q \subseteq \mathrm{Pos}\setminus\{p\}$ to be substituted by variables. The next result shows that for these, there are only two possible outcomes.

Lemma 2.8. Let F, I and p be as above. For any $Q \subseteq \mathrm{Pos}\setminus\{p\}$ we have $\mathrm{INF}_I(p \mid F, Q) \in \{0, 1\}$.

Proof. Consider any "fresh" value $a \in \mathbb{N}$, which does not appear in the column of position p in the relation instance I. It is easy to see that, whether the instance $(I_{Q\leftarrow X})_{p\leftarrow a}$ fulfills F or not, does not depend on the choice of those values a. Therefore, as $k \to \infty$ we either have $\log \#V_Q / \log k \to 1$ or $\log \#V_Q / \log k \to 0$. □

From this lemma and its proof, we deduce the following simplification for computing information content.

Proposition 2.9. Let F be a set of functional dependencies and I an instance with $I \models F$. Let p be a position in I with attribute A, and let $a \in \mathbb{N}$ be any value that does not appear in the column of attribute A. This yields:

$$\mathrm{INF}_I(p \mid F) = \frac{\#\{Q \subseteq \mathrm{Pos}\setminus\{p\} \mid (I_{Q\leftarrow X})_{p\leftarrow a} \models F\}}{2^{\#\,\mathrm{Pos}\,-1}}$$

Although we have simplified the computation of the information content somewhat, one still needs to consider all subsets of $\mathrm{Pos}\setminus\{p\}$, leading to exponential complexity. Therefore, we present in the following several optimizations for this computation.

3. Optimizations

In this section, we provide optimizations to speed up the computation of information content. In Section 3.1 we deal with exact methods and prove their correctness, while in Section 3.2 we present an approximation.

3.1. Reducing the Problem Size

We give two shortcuts for computing the information content in a relation instance. The first identifies the positions where there cannot be any redundancy, so that the information content equals 1.

Definition 3.1. Let $p = (j, B) \in \mathrm{Pos}$ be a position with attribute B in an instance I and let f be a functional dependency $A_1 \ldots A_s \to B$ with $I \models f$. We say that the value at p is *unique* with respect to f if for every $j' \in \mathrm{Def}(I)$ there holds

$$t_j[A_1 \ldots A_s] = t_{j'}[A_1 \ldots A_s] \;\Rightarrow\; j = j'$$

(so if $j \neq j'$, then $t_j[A_1 \ldots A_s] \neq t_{j'}[A_1 \ldots A_s]$).

For a set F of functional dependencies with $I \models F$, the value at p is *unique* with respect to F if it is unique with respect to all $f \in F$ of the form $A_1 \ldots A_s \to B$.

Note that in particular, a value at position p with attribute B is unique with respect to F in case the attribute B does not appear on the right-hand side of any functional dependency in F.

Proposition 3.2. Let $p \in \mathrm{Pos}$ be a position in an instance I, where $I \models F$ for a set of functional dependencies F. Then $\mathrm{INF}_I(p \mid F) = 1$ if and only if the value at p is unique with respect to F.

Proof (of the "if" direction). Let B be the attribute of position p and let a be a "fresh" value not appearing in the column of attribute B in the instance I. By Prop. 2.9 it suffices to show that for every subset $Q \subseteq \mathrm{Pos} \setminus \{p\}$ it holds that $(I_{Q \leftarrow X})_{p \leftarrow a} \models F$. So let $f \in F$ be a functional dependency $A_1 \dots A_s \to B'$. From $I \models f$ we know that $I_{Q \leftarrow X} \models f$. Then we have to show that

$$(I_{Q \leftarrow X})_{p \leftarrow a} \models f,$$

i.e., if $t_{j_1}[A_1 \dots A_s] = t_{j_2}[A_1 \dots A_s]$ for some $j_1, j_2 \in \mathrm{Def}(I)$ it still holds that $t_{j_1}[B'] = t_{j_2}[B']$, even after inserting value a at position p.

Suppose first that $B \neq B'$. If $B \notin \{A_1, \dots, A_s\}$ (and $B \neq B'$) the assertion above is clear for any value a, since the statement is not affected. On the other hand, if $B = A_i$ for some $1 \leqslant i \leqslant s$, then by inserting the fresh value a the hypothesis becomes false, so the statement remains valid.

Now consider the case $B = B'$ and let $p = (j, A_k)$. We may assume that one of j_1, j_2 equals j and write j' for the other index. Then if $t_j[A_1 \dots A_s] = t_{j'}[A_1 \dots A_s]$, then by the uniqueness property we infer that $j = j'$. So the above statement still holds after inserting the value a, since the tuple indices coincide. □

In the second optimization, we reduce the considered instance to the relevant tuples and attributes. This may reduce the number of cells, and thus decrease the runtime exponentially. After performing this step, we can apply the first shortcut in the smaller table and use the outcome for the original instance.

Let I be a relation instance of arity m and let $J \subseteq \mathrm{Def}(I)$, $K \subseteq \{A_1, \dots, A_m\}$ where the A_k are the attributes of the relation. The subinstance $I(J, K)$ consists of all tuples of the projection $\pi_K(I)$ with index $j \in J$. We also let $\mathrm{Pos}(J, K) := J \times K \subseteq \mathrm{Pos}$ be the corresponding set of positions.

Proposition 3.3. Let F be a set of functional dependencies and I an instance with $I \models F$. Denote by J_0 the set of all indices $j_0 \in \mathrm{Def}(I)$ such that for some position $p = (j_0, A_k) \in \mathrm{Pos}$ the value at p is not unique with respect to F. Denote by K_0 the union of all attributes $\{A_1, \dots, A_s, B\}$ involved in any functional dependency $A_1 \dots A_s \to B$ in F. Then for every $J \supseteq J_0$ and $K \supseteq K_0$ there holds

$$\forall p \in \mathrm{Pos}(J, K) \colon \ \mathrm{INF}_I(p \mid F) = \mathrm{INF}_{I(J,K)}(p \mid F).$$

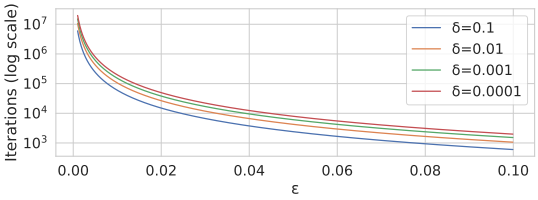


Figure 2: Required iterations to achieve an accuracy (ε) with a certain confidence ($1-\delta$) in Monte Carlo approximation.

Example 3.4. Consider the instance

A	B	C	D
7	2	8	4
5	2	8	6
7	2	8	6

and the set of functional dependencies $F := \{\mathrm{A} \to \mathrm{C}\}$.

Since the attributes A, B or D do not appear on the right-hand side of the functional dependency, Prop. 3.2 implies that $\mathrm{INF}_I(p \mid F) = 1$ for all $p = (j, A_k)$ with $A_k \neq \mathrm{C}$. Additionally, by Prop. 3.2 we obtain that $\mathrm{INF}_I((2, \mathrm{C}) \mid F) = 1$, since the value at position $p = (2, \mathrm{C})$ is unique with respect to F.

We can reduce the table using Prop. 3.3 by removing the second tuple and the attributes B and D. The resulting subtable is

A	C
7	8
7	8

for which the number of subsets in $\mathrm{Pos}(J, K) \setminus \{p\}$ is reduced from 2^{15} to 2^3, i.e., by a factor over 4'000.

Consider position $p = (1, \mathrm{C})$ and the subsets $Q \subseteq \mathrm{Pos}(J, K) \setminus \{p\}$. For $Q = \varnothing$, i.e., the values in all positions different from $(1, \mathrm{C})$ are present, the instance $(I_{Q \leftarrow X})_{p \leftarrow v} = I_{p \leftarrow v}$ fulfills the functional dependency $\mathrm{A} \to \mathrm{C}$ only if $v = 8$. For all other subsets Q, i.e., at least one more value is lost, we obtain $(I_{Q \leftarrow X})_{p \leftarrow v} \models F$ for all values v. Therefore, $\mathrm{INF}_I(p \mid F) = \frac{7}{8} = 0.875$. The computation of $\mathrm{INF}((3, \mathrm{C}) \mid F)$ is similar, and we get

$$\big(\mathrm{INF}_I((j, A_k) \mid F)\big) = \begin{pmatrix} 1 & 1 & 0.875 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 0.875 & 1 \end{pmatrix}.$$

3.2. Monte Carlo Approximation

Next, we present an approximation of the information content. This approach is complementary to the previous optimizations for computing exact values, e.g., by identifying cells with full information content first, then

reducing the problem to a smaller subtable (if possible), and finally computing the values on this subtable with an approximation algorithm.

The approximation is computed with a randomized algorithm, the Monte Carlo method, as introduced in [5]. Instead of considering all subsets of positions (minus the considered one), we pick a sample of subsets uniformly at random and take the average of the results. The following tool is deduced from Theorem 4.14 of [5] and is useful for estimating the accuracy of this method.

Proposition 3.5 (Hoeffding's inequality). Let $X_1, \ldots, X_n$ be independent identically distributed random variables such that $a_i \leqslant X_i - \mathrm{E}[X_i] \leqslant b_i$. Then for any $c > 0$ it holds

$$\Pr\Big(\sum_{i=1}^{n}(X_i - \mathrm{E}[X_i]) \geqslant c\Big) \leqslant \exp\Big(\frac{-2c^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\Big).$$

To approximate the information content, we consider the probability space $\mathcal{B}(I,p) := (\mathcal{P}(\mathrm{Pos}\setminus\{p\}), P_{\mathcal{B}})$ from Section 2 with uniform distribution $P_{\mathcal{B}}$. Define random variables

$$X_i \colon \mathcal{P}(\mathrm{Pos}\setminus\{p\}) \to [0,1], \quad Q \mapsto \lim_{k\to\infty} \frac{\log \#V_Q}{\log k},$$

with $V_Q := \{v \in \{1,\ldots,k\} \mid (I_{Q\leftarrow X})_{p\leftarrow v} \models F\}$. Then by Prop. 2.9 there holds $\mathrm{E}[X_i] = \mathrm{INF}_I(p \mid F)$.

This motivates the next theorem on the randomized approach to compute the information content with accuracy ε and confidence $1-\delta$.

Theorem 3.6. Let $p \in \mathrm{Pos}$ be a position in a relation instance I, where $I \models F$ for a set F of functional dependencies. Let $X_1, \ldots, X_n$ be independent identically distributed random variables as above and $X := \frac{1}{n}\sum_{i=1}^{n} X_i$ their average. Then for all $\varepsilon, \delta > 0$ it holds

$$\Pr\big(|X - \mathrm{INF}_I(p \mid F)| \geqslant \varepsilon\big) \leqslant \delta$$

provided that $n \geqslant 2\ln(2/\delta)/\varepsilon^2$.

Proof. We have $-1 \leqslant X_i - \mathrm{E}[X_i] \leqslant 1$, so by applying Hoeffding's inequality with $\sum_{i=1}^{n}(b_i - a_i)^2 = 4n$ we find for $\Pr(\sum_{i=1}^{n}(X_i - \mathrm{E}[X_i]) \geqslant n\varepsilon)$ the upper bound $\exp(-n\varepsilon^2/2)$, so that $\Pr(|X - \mathrm{E}[X_i]| \geqslant \varepsilon) \leqslant 2\exp(-n\varepsilon^2/2) \leqslant \delta$. □

Example 3.7. Assume that for the approximation of information content for an instance I at position p with respect to F, we allow an error of at most 0.001 with probability at least 99.9%. This can be achieved by sampling $Q \subseteq \mathrm{Pos}\setminus\{p\}$ and computing $X_i(Q)$ at least $2\ln(2/10^{-3})/(10^{-3})^2 \geqslant 1.52{\cdot}10^7$ times.

If a less exact approximation is sufficient, say with an error of at most 0.01 with the same probability as before, then the required number of runs is lowered by a factor 100, thus only $1.52{\cdot}10^5$ samples are necessary. Figure 2 shows a plot of the iterations required for reaching a certain accuracy (ε) and a certain confidence ($1-\delta$).

4. Experiments

Our experiments target the following research questions:

RQ1 Is the plaque test useful for real-world datasets?

RQ2 Can we afford to compute *exact* entropy values or do we need the Monte Carlo approximation?

RQ3 How does the runtime of the Monte Carlo approximation scale with the number of iterations?

In the following, we report on our insights with a first prototype and three real-world datasets.

Implementation Our prototype implements our algorithms as a single-threaded Java implementation. As a dispatcher, we use a Python script that also measures the end-to-end runtimes.

Datasets We explore three real-world datasets. All functional dependencies are left-reduced with a single attribute on the right and were discovered by Metanome [4].

Our first dataset describes natural satellites and originates from the WDC Web Table Corpus[3]. Metanome finds 35 functional dependencies, we analyze the first 150 rows. We study two additional datasets[4]: The adult dataset (where we analyze the first 150 rows) with 78 functional dependencies captures census data. The echocardiogram dataset (all 132 rows) with 538 functional dependencies describes heart attack patients.

Environment Our server has an Intel Xeon Gold 6242R (3.1 GHz) CPU and 192GB of RAM, and runs Ubuntu 22.04 with Java 18.

Results We now address the research questions in turn.

RQ1 We computed the plaque tests for the three datasets and show the results in Figure 3. Entropies are computed with Monte Carlo simulation set to 100'000 iterations, an accuracy of approx. 0.01, and a confidence of 99%.

Cells with an entropy value of 1 are shown in white, and cells with values below 1 are colored in blue, where darker shades indicate lower entropy values. Color scales are calibrated individually, so colors cannot be compared between datasets. We discuss each dataset.

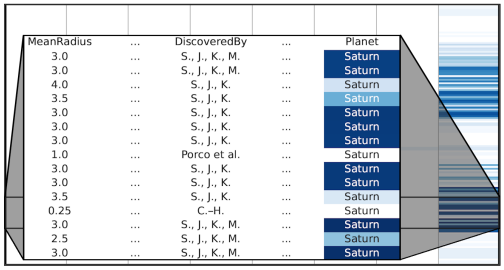
Satellites. Figure 3a shows the plaque test applied to the satellite dataset. Plaque is locally concentrated, as only the column "Planet" and very few cells in column "Notes" contain entropy values below 1.

We zoom in on a subset of the rows, omitting cells with an entropy value of 1 and showing the values for columns "MeanRadius", "DiscoveredBy", and "Planet".

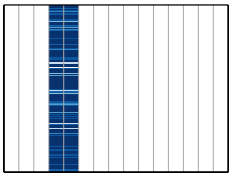
For tuples with a mean radius of 3.0, the entropy of the attribute planet is the lowest. A mean radius of 3.0 occurs only for Saturn satellites. Closer inspection reveals that "MeanRadius" is on the left-hand side of several functional dependencies (as discovered by Metanome), including "MeanRadius, DiscoverdBy → Planet".

[3] http://webdatacommons.org/webtables/index.html#results-2015

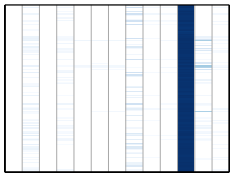
[4] Retrieved from https://hpi.de/naumann/projects/repeatability/data-profiling/fds.html, on June 02, 2023



(a) Satellites (150, Min: 0.61)



(b) Adult (150, Min: 0.50)



(c) Echoc. (132, Min: 0.00)

Figure 3: "Plaque tests" applied to real-world data. The subcaptions state the numbers of rows analyzed and minimum entropy values computed (rounded). The color scale is normalized individually with respect to the minimum entropy value. The zoom-in in Subfigure (a) highlights a subset of rows.

By inspecting the causes for low entropy values in this fashion, data analysts can gain insight into why automated tools discover certain dependencies. They can then decide whether the dependency is genuine or merely an artifact of the data.

Adult. Figure 3b shows the plaque test applied to the census data. Only two columns, "education" and "education-num" have entropy values below 1. Moreover, in each row, both columns have the *same* entropy value. Closer inspection reveals that there are functional dependencies "education-num → education" as well as "education → education-num". This causes the respective entropy values to agree.

In consequence, a data analyst might decide to decompose this relation into the second normal form, by storing the mapping between "education" and "education-num" in a separate relation.

In fact, normalization theory was a main motivator behind the original work by Arenas and Libkin.

Echocardiogram. Figure 3c shows the plaque test applied to the patient data. Among the three datasets, this dataset has the highest number of columns with entropy values below 1: This affects 11 out of 13 columns, but mostly only sparsely.

One column stands out, where all entropy values are zero. Inspection reveals that this is the column that originally contained the patient's name, which was changed to a single global string constant as a means of anonymization. Consequently, for every attribute, there is an obvious functional dependency, with this attribute on the right-hand side. Our plaque test correctly reveals that this column literally has no informational value.

Table 1
Runtimes in seconds for computing exact entropy values w/ and w/o the optimizations from Section 3 for the first i rows of satellite data. Runs marked "-" were aborted after 24 hours.

#Rows	Unoptimized	Optimized
1	0.128	0.097
2	1.318	0.099
3	461.059	0.320
4	-	0.355
5	-	25'221.186
6	-	-

Results. We applied our visual plaque tests to standard datasets used in dependency discovery research. The plaque tests appear to be helpful in data exploration: When "plaque" is detected, we can always find an intuitive explanation for its causes.

In the examples discussed, it highlighted a particularly prominent functional dependency, it revealed a good opportunity for schema normalization, and it exposed data with no informational value.

Moreover, the plaque test is very selective: The test is strongly positive for only a few attributes. This effect is also observable for two further datasets that we analyzed in the long version of this article.

This sparsity of cells that test strongly positive for plaque is particularly notable in the case of the echocardiogram dataset, despite the over 500 automatically discovered dependencies. Compared to this high number of dependencies, the result of the plaque test is easily consumable, and data analysts are visually directed towards the most pertinent redundancies.

RQ2 Table 1 shows the runtime in seconds for computing the entropy values on subsets of the satellite data. We compute exact entropies and do not yet apply the Monte Carlo approximation. We compare the algorithm with the optimizations from Section 3 disabled/enabled.

The unoptimized algorithm can process only three rows in 24 hours. Using the optimizations, up to five rows could be computed in 24 hours.

Results. Although the optimizations are effective, such runtimes are still too slow for practical purposes.

RQ3 We next explore the Monte Carlo approximation, in combination with our optimizations from Section 3.

Figure 4 shows the runtimes in seconds for different subsets of the satellite data and different numbers of Monte Carlo iterations.

For reasonably large subsets, runtime scales linearly with the number of iterations, while input size influences runtime more heavily. Calculating the entropies for 150

Number of Rows	Monte Carlo Iterations 1000	10'000	100'000	1'000'000
10	0.3	0.9	5.2	45.2
40	1.5	9.1	82.6	793.4
70	4.1	38.0	375.9	3'571.7
100	7.7	71.9	656.0	6'526.3
130	13.5	123.9	1'236.2	12'279.1
150	18.0	166.7	1'646.8	16'051.4

Figure 4: Runtime in seconds on the satellite dataset, for different numbers of Monte Carlo iterations and subset sizes of satellite data. Higher saturation indicates longer runtimes.

rows takes around 4.5 hours at 1'000'000 iterations. However, we reach an accuracy of approx. 0.01 with 99% confidence at 100'000 iterations (see Figure 2), which takes around 30 minutes to calculate.

Results. The approximation greatly improves the runtime behavior. Since we use the entropy values as a basis for visualization, small deviations in accuracy lead to equally small deviations in the color scale. The differences will most likely not be discernible to the human eye.

However, scalability to larger inputs remains a challenge and will require further improvements.

5. Related Work

Dependency discovery is an established and active field, and we refer to [6] for an overview. Visualizing dependencies is not as well studied, and existing approaches such as sunburst diagrams or graph-based visualizations [2] do not take the data instance into account. On the contrary, our plaque test does not visualize the dependencies per se, but the redundancies captured by them in the data.

In our visualization, we impose heat maps over relational data. Heat maps have also been explored elsewhere, such as to visualize the frequency of updates [7].

We build on an information-theoretic framework developed by Arenas and Libkin [3], which aligns with classic normalization theory. Notably, we are not aware of any earlier implementations of this framework.

In an orthogonal effort, Lee [8] proposed entropies at the instance level which are not tied to a set of functional dependencies. Therefore, this approach does not lend itself to the plaque test proposed here.

6. Outlook

We propose a plaque test to visualize redundancies in relational data, based on functional dependencies. As our discussion of real-world examples shows, the presence of plaque reveals interesting redundancies in the data. As we have seen, the plaque may single out prominent dependencies, reveal the need for schema normalization, or expose data as meaningless.

In our experiments, we work with data sets of only a modest size. Scaling to larger datasets is a challenge for future work, and we see optimization opportunities via parallelization or linear programming.

Once we are able to scale to larger datasets, we will need to address the consumability of our visualization. For example, we may allow users to interactively cluster cells with plaque for easy browsing and inspection.

We may allow users to examine the effect of individual dependencies by excluding them from the visualization in an exploratory manner. This raises the question of whether the entropies can be computed incrementally.

We also plan to investigate the visualization of other kinds of dependencies, such as join dependencies, since they are also covered by the underlying information-theoretic framework.

Acknowledgments This work was partially funded by *Deutsche Forschungsgemeinschaft* (DFG, German Research Foundation) grant #385808805. We thank Meike Klettke for her feedback on an earlier version of this article.

References

[1] S. Abiteboul, R. Hull, V. Vianu, Foundations of Databases, Addison-Wesley, 1995.

[2] S. Kruse, D. Hahn, M. Walter, F. Naumann, Metacrate: Organize and Analyze Millions of Data Profiles, in: Proc. CIKM, 2017.

[3] M. Arenas, L. Libkin, An information-theoretic approach to normal forms for relational and XML data, in: Proc. PODS, 2003.

[4] T. Papenbrock, T. Bergmann, M. Finke, J. Zwiener, F. Naumann, Data profiling with metanome, Proc. VLDB Endow. 8 (2015).

[5] M. Mitzenmacher, E. Upfal, Probability and Computing: Randomization and Probabilistic Techniques in Algorithms and Data Analysis, Cambridge University Press, 2017.

[6] Z. Abedjan, L. Golab, F. Naumann, T. Papenbrock, Data Profiling, Synthesis Lectures on Data Management, Morgan & Claypool Publishers, 2018.

[7] T. Bleifuß, L. Bornemann, D. V. Kalashnikov, F. Naumann, D. Srivastava, Dbchex: Interactive exploration of data and schema change, in: Proc. CIDR, 2019.

[8] T. Lee, An Information-Theoretic Analysis of Relational Databases—Part I: Data Dependencies and Information Metric, IEEE Transactions on Software Engineering SE-13 (1987).