DataSense: Display Agnostic Data Documentation

Don't know what to make of your data? Ask DAD for help!

Poonam Kumari, Michael Brachmann, Oliver Kennedy
University at Buffalo, SUNY
{poonamku,mrb24,okennedy}@buffalo.edu

Su Feng, Boris Glavic
Illinois Inst. of Technology
{sfeng14@hawk,bglavic@}.iit.edu

ABSTRACT

Documentation of data is critical for understanding the semantics of data, understanding how data was created, and for raising awareness of data quality problem, errors, and assumptions. However, manually creating, maintaining, and exploring documentation is time consuming and error prone. In this work, we present our vision for display-agnostic data documentation (DAD), a novel data management paradigm that aids users in dealing with documentation for data. We introduce DataSense, a system implementing the DAD paradigm. Specifically, DataSense supports multiple types of documentation from free form text to structured information like provenance and uncertainty annotations, as well as several display formats for documentation. DataSense automatically computes documentation for derived data. A user study we conducted with uncertainty documentation produced by DataSense demonstrates the benefits of documentation management.

1 INTRODUCTION

Producing and consuming **documentation** is an essential step when creating and analyzing data. Good documentation helps a consumer of data to understand the context of the data and its schema, including (i) semantics (e.g., the currency of an account balance), (ii) data collection techniques (e.g., which assay was used to measure blood iron), (iii) limitations or caveats (e.g., that missing values are due to sensor failure), and (iv) assumptions made (e.g., missing geographical locations were inferred through geocoding). Misunderstanding the context of a dataset can lead to statistical errors and mistakes with potentially serious, life-threatening consequences. In short, good data documentation is critical. Unfortunately, extensive documentation can be overwhelming making it hard to find elements in the documentation that are relevant for the user's task at hand. We need a better way to interact with documentation than the current state of the art: dozens or even hundreds of pages of word documents.

Example 1. *The NYC Taxi Trip dataset [8] is a comprehensive catalog of taxi travel in NYC, and is used to provide insights into many aspects of city life, human behavior, and economic activity. The dataset also has numerous outliers, many resulting from legitimate events [12] like hurricanes and holidays, while others are irreparable data errors (trips starting in Hudson river). On joining a lab working with this dataset, Alice discovers that her new lab-mates have been diligently maintaining a document cataloging outliers and other considerations for working with the dataset. She reads the document, but misses the reminder that a hurricane hit NYC in August, 2011, and spends the next several days trying to explain a significant dip in August travel.*

Conference'17, July 2017, Washington, DC, USA
2021. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

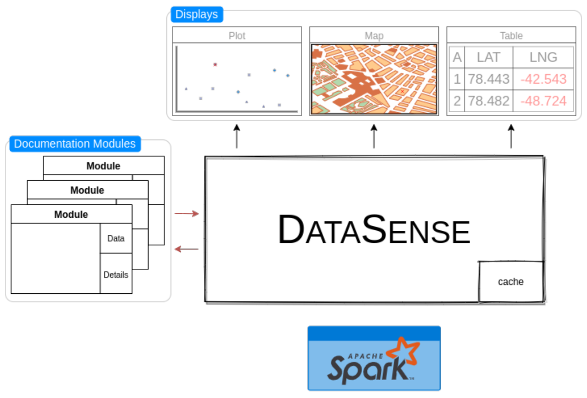


Figure 1: The DataSense abstraction layer

Taking inspiration from modern code development environments (IDEs), we propose **display-agnostic data documentation** (**DAD**), a novel data interaction paradigm that facilitates both *discovery* and *lookup* of context-relevant data documentation by inlining them directly and *interactively* into the data display. This documentation can often be derived from the data or by analyzing provenance. Indeed, many such techniques already exist, ranging from fully automated data documentation techniques like data profiling [1] and provenance summarization [2, 19], to user-provided prose annotations [9]. Consider the following examples of useful documentation:

Outliers: Highlighting outliers in a dataset can help users to find errors or interesting data points. Additional textual documentation can be used to explain them.

Missing Values: Under SQL's `NULL` Semantics, aggregates silently ignore null values. Highlighting data derived from null values can reveal data errors like failed `CAST` operations that might invalidate analyses based on the data, or unreliable aggregated results computed mostly from `NULL` values.

Cell Provenance: Spreadsheet expression cells (e.g., `'=$A22+$B22'`) can help users to interpret the role of the cell, for example when the cell's column name is uninformative. Similar information for database query results (e.g., `AVERAGE(ST_Distance(trip.start, trip.end))`). i.e., schema-level provenance, can be just as helpful.

Annotations: Semistructured documentation is used in programming languages like Python (`__doc__` or PEP484) and Java (Javadoc), and leveraged by IDEs for mouse-over contextual documentation.

Annotation management [5, 9] explores how user-generated semistructured documentation can be propagated through queries. Annotations can be incredibly valuable, for example, to identify data collected under unusual conditions (e.g., extreme weather events).

Example 2. *Consider an alternate universe where Alice's lab-mates have written their documentation in the form of semi-structured prose that can be scraped by a hypothetical DAD system. For example, in a notebook like Jupyter or Vizier*[1]*, such details can be added as specially formatted markdown cells. When Alice views the number of trips aggregated by month, the count for August is now highlighted. She mouses over the month and sees (among other things) a note saying that this count might be affected by the hurricane that month, saving her days of data exploration.*

Unlike IDEs where users interact with code only through a text editor, a dataset may be displayed and accessed through many modalities: as a table, in a graph, on a map, by writing queries, and more. The need to support multiple modalities poses a challenge for inline documentation, as each new form of data documentation needs to be individually adapted for each display modality. We propose an abstraction (DAD) to separate logic for generating data documentation, from the logic for displaying data with the documentation inlined. A standard interface between these components makes it possible to implement a documentation generator once, and have it work with a range of display modalities. We have four central goals for the design of the DAD abstraction:

Process Agnostic: DAD aims to support a variety of metadata including both free-form prose and structured properties extracted from the data, associated metadata, and the query that generated it. DAD needs to make minimal assumptions about the process generating the documentation.

Declarative: Documentation generated for DAD should be agnostic to the modality in which it will be displayed. We prefer the use of declarative roles (e.g., highlight, detail) to explicit formatting rules.

Context Sensitive: While extensive documentation is extremely valuable, it can also be overwhelming. Thus it is important to dynamically adapt documentation to what is relevant for a user.

Unobtrusive: DAD should (i) facilitate *discovery* of relevant documentation without impeding the user's normal data interactions, while also (ii) making context-relevant documentation accessible.

In this paper, we focus on DataSense, a concrete implementation of the DAD abstraction illustrated in Figure 1. DataSense provides an extensible interface through which *documentation modules* produce content that can be incorporated into the display of a dataset by one or more *display managers*. The first major design question is what types of documentation should be shown. Inspired by IDEs, and in keeping with the above goals, DataSense provides two broad classes of documentation displays. To facilitate discovery, DAD allows documentation modules to mark elements of the dataset with declarative *display classes*. These identify what makes the element interesting, but leave presentation issues to the display manager. To facilitate access to context-relevant documentation, DataSense allows documentation modules to contribute content to a detail view describing a specific dataset element.

[1]Jupyter: https://jupyter.org, Vizier https://vizierdb.org

Rather than developing specialized interfaces for each form of documentation, DataSense's display managers only need to consider how to present highlights and detail views. For instance, the tabular display manager interprets display classes as color highlights, and reveals detail views when an element is clicked. Conversely, the plot display manager interprets display classes as icons, and reveals detail views when an element is moused over. We note that highlights and detail views can also find use in existing IDE documentation frameworks. For instance, when writing a SQL query, we may show documentation for column names that the user's mouse is hovering over (e.g., this column has 15% outliers).

Concretely, DataSense extends Apache Spark-Scala with support for a range of display operations that draw on an extensible library of documentation modules. We explore the design of DataSense in Section 2. Section 3 shows, through a user study, that this approach is effective in making relevant documentation available to users in a digestible form that helps them to better understand limitations of the data they are using in their analysis. We cover related work in Section 4 and conclude with remaining challenges in Section 5.

2 DATASENSE

As shown in Figure 1, DataSense is an abstraction layer between *documentation modules* that generate contextual documentation, and relational data *display managers* that render (e.g., as a table, plot, or map) the data and its documentation. The goal of DataSense is to remain unobtrusive, while still facilitating (i) documentation discovery, and (ii) detailed, contextual documentation. Thus, two forms of documentation are supported: (i) low bandwidth data highlights that communicate simple data features in line with the data, and (ii) high bandwidth detail views that communicate more complex information. Highlights and detail views may be defined for cells, columns, rows, or the dataset as a whole.

DataSense is implemented over Spark-Scala Dataframes. Spark is a popular distributed data processing framework with a Dataframe abstraction analogous to relational tables, albeit for imperative programming languages. Spark Dataframes expose the standard array of relational operators through method calls, and internally track transformations symbollically as a logical query plan. When the Dataframe's contents are required, it must be executed (i.e., compiled and deployed to a cluster). Spark is designed for interactive use (e.g., in a notebook): Spark caches intermediate results to reduce the cost of iteratively refined queries, and provides convenient accessors for displaying or summarizing the data. Between first-class support for interactive analytics and the fact that Dataframes retain symbolic representations of their workflow provenance, Spark Dataframes are an ideal target for DataSense.
To recap, Figure 1 shows the three types of components:

(1) Documentation modules responsible for creating contextual documentation for a relational table.
(2) Display managers responsible for rendering a relational table and its documentation (e.g., as a scrollable table or a plot).
(3) The coordinating DataSense hub.

We now detail these components, first the hub's interactions with documentation modules to efficiently produce the two supported forms of documentation, and then with the display manager.

2.1 Highlighting Data

To facilitate discovery, DataSense allows documentation modules to mark specific data elements — rows, columns, cells, or the dataset itself — for highlighting. As in many IDEs, documentation modules mark data elements with HTML-like display classes like `notForRelease`, `outlier`, or `annotated`. Each module exports a set of display classes that it supports and suggests rendering options for each, including symbols to render over highlighted points on plots and CSS styles to use for highlighted elements in tabular views. We now explore three examples of highlighting.

Export Control: For users working with data subject to varying levels of privacy restrictions, one useful form of documentation is the level of secrecy required for a given dataset: either public or not-for-release. A simple implementation uses a metadata store to identify protected data sources, exports a single `notForRelease` display class, and marks any Dataframe if a protected data source appears anywhere in its lineage. Computing markings only requires static analysis and a small number of queries to the metadata store.

Outliers: Identifying outliers can help users to find errors or interesting data points. A simple implementation exports a single `outlier` display class, and highlights numerical values that deviate from the mean by more than two standard deviations. Computing markings requires means and standard deviations, which may require additional queries.

Annotations: User-generated prose can warn users of a dataset about unusual dataset features like extreme weather events during the period in question. A simple implementation maintains annotations in an external table, exports a single `annotated` display class, and highlights rows or cells derived from an annotated value to alert the user to the annotation's presence. Computing markings requires an instrumented version of the original Dataframe query.

Naively, DataSense would (i) execute the Dataframe, (ii) compute display classes, and (iii) pass both results to the display manager responsible for rendering. Executing these steps in sequence is slow, particularly when additional queries are required. A simple mitigation is to complete steps (i) and (ii) in parallel. For example, the mean and standard deviation of a Dataframe's numerical columns can be computed in parallel with the Dataframe itself.

In some cases, DataSense can do better by inlining data marker computation directly into the Dataframe query. Before the Dataframe is executed, each documentation module is given the opportunity to rewrite it, extending the Dataframe with additional columns. It is the documentation module's responsibiltiy to avoid modifying the rows or columns present in the original dataframe.

2.2 Detail Views

Documentation modules also produce contextual documentation to be displayed, for example, when a user clicks on a table cell or header, or mouses over a point in a plot. When a detail view is requested, the DataSense hub forwards the request to all display modules in parallel. Each documentation manager responds with an HTML5 representation of the detail. We explore three use cases:

Semantic Types: More precise types that incorporate contextual details like units or method of collection can be useful for understanding the context of a column or debugging errors. The types of attributes can either be automatically inferred [14, 21], or derived through static analysis (e.g., $\frac{\text{distance:km}}{\text{time:hours}} = \text{speed:km/hr}$). If automatic discovery happens offline, only static analysis is needed.

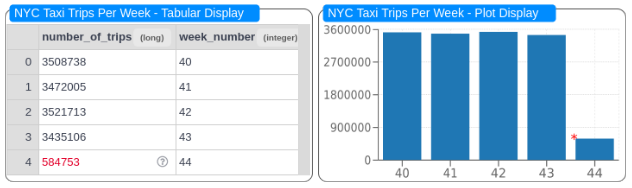
Cell Provenance: Displaying the provenance of a cell can help users to better understand it. As noted already, provenance can be summarized into a compact representation for easy display. However, at the granularity of cells, obtaining detailed provenance for certain operators like group-by or union requires running an instrumented version of the query (e.g., to determine the aggregate group from which the cell was derived).

Missing Values: Under SQL's `NULL` Semantics, aggregates silently ignore null values. This can obscure data errors like failed `CAST` operations, potentially invalidating analyses based on the data. The number and ratio of `NULL` to non-`NULL` values used to compute a data point can be helpful, but requires an instrumented query.

DataSense mitigates the high cost of constructing detail views in 3 ways: (i) Constructing detail views asynchronously and in parallel, (ii) Caching detail views, and (iii) Using query rewriting (as for highlighting) to pre-compute content for detail views.

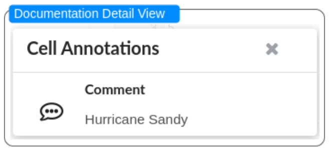
2.3 Display Manager

To render a Dataframe, the DataSense hub invokes the corresponding display manager with the points to be displayed and the computed display classes. A unique name for each column and an assigned identifier for each row allows individual columns, cells, and rows to be referenced. When a detail view is requested by the display manager, the DataSense hub returns futures for the detail view to be displayed for each documentation module. The display manager displays the detail view immediately, and monitors the futures; As each future completes, the resulting HTML is displayed.


NYC Taxi Trips Per Week - Tabular Display

	number_of_trips (long)	week_number (integer)
0	3508738	40
1	3472005	41
2	3521713	42
3	3435106	43
4	584753	44

Figure 2: NYC Taxi Data Displayed


Documentation Detail View

Cell Annotations

Comment

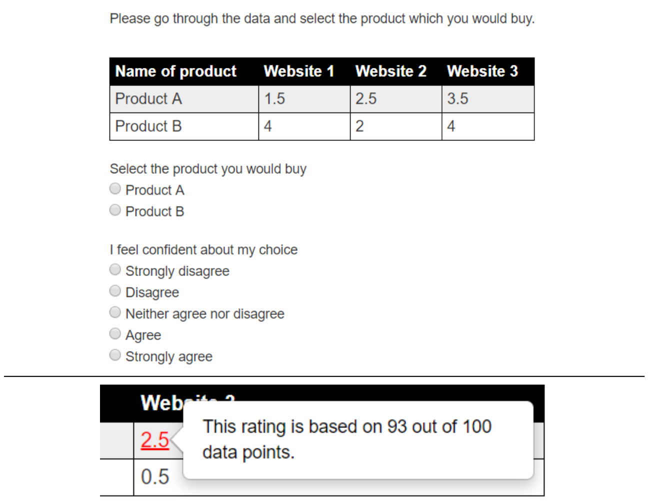
Hurricane Sandy

Figure 3: NYC Taxi Detail View

3 CASE STUDY: MISSING VALUES

We explore DataSense's discovery/detail approach through a case study based on aggregate values affected by missing data: A user is tasked with making a decision, and given aggregate summary statistics in tabular form to base their decision on. The catch is that the source data contains missing (i.e., `NULL`) values.

Missing values make aggregate results imprecise, though not necessarily wrong: if only a few values are missing, users may still *trust* the result. However, the choice of whether or not to trust the data is subjective — A human must be in the loop for this decision.



Please go through the data and select the product which you would buy.

Name of product	Website 1	Website 2	Website 3
Product A	1.5	2.5	3.5
Product B	4	2	4

Select the product you would buy
- Product A
- Product B

I feel confident about my choice
- Strongly disagree
- Disagree
- Neither agree nor disagree
- Agree
- Strongly agree

2.5

This rating is based on 93 out of 100 data points.

0.5

Figure 4: User Interface for Online Experiment

While we need to make the user aware of uncertainty through documentation, we want to avoid burdening the user with work for irrelevant values — values that appear in the display but that are not of interest to the user. The idea is to highlight values to indicate that they depend on null values, and on click provide a detail view as introduced in Section 2.2, which shows the total size of the highlighted value's provenance and the number of null values in the provenance. If the fraction of null values in the provenance is low in the user's opinion, then the value can be considered trustworthy, while if the fraction is high then the value is untrustworthy.

The Missing Value Documentation Module: This module instruments queries to flag `NULL` values in input sources: The schema is extended with a {0,1}-valued integer marker for each attribute of the query, which is 1 for a row if the row's value in this attribute is `NULL` and 0 otherwise. During query execution, counters are propagated for each such attribute to track the number of null values a cell depends on, e.g., for aggregates and `DISTINCT`, the counters for all inputs contributing to a result are summed up (e.g., as in [5, 10]) . The total number of input values on which an output depends on is computed similarly. Data values with non-zero null-counters are highlighted, and the cell detail view displays both the non-null-count and total dependents count.

3.1 User Study

We designed a four-variable mixed study for Amazon Mechanical Turk with a one-variable (3-way) between-subject component and a three-variable ($7 \times 2 \times 2$-way) within-subject component. The study was approved by the authors' university institutional review board (IRB). We simulate low-impact decision-making based on tabular data. A total of 900 participants (319 female, 575 male and 6 undisclosed gender) with a median age from 18 to 29 participated.

Stimuli & Tasks: We generated a fixed set of 56 2x3 tables and presented them in a random order through the interface illustrated in Figure 4. Participants were asked to select one of two products for purchase based on three ratings on a scale from 0 to 5, and were informed that each table contained summary ratings from 3 retailer's websites. We measured three variables for each trial: (1) The time taken, (2) The product choice, and (3) Whether the participant requested a detail view.

Each table was designed to measure one combination of the three within-subject variables: `Relevance` (2-way), `Trustworthiness` (2-way), and `Highlight Format` (7-way). We used rejection sampling to select tables that would suggest a specific product choice as the better product (randomized for each table), with one significantly different (1 point difference or more) rating pair and two approximately similar (less than 1 point difference) rating pairs. For instance, in Figure 4, Product B should be chosen as the better product, since the rating from website 1 is extremely favorable, the website 3 rating is marginally favorable, and the website 2 rating is marginally disfavorable. For each table, we randomly selected a *designated highlight* value: One of the two significantly different values (`Relevance=1`, or "Relevant"), or one of the four insignificantly different values (`Relevance=0`, or "Irrelevant"). The rationale is that if one value of a rating pair which is similar for both products is uncertain, then this is irrelevant for the decision since another rating pair for which both values are certain determines which product should be preferred. However, if one of the values of the rating pair with a significant difference is uncertain then this uncertainty is relevant for the decision.

For each table, we also generated detail view text of the form "This value is an average based on X of 100 source values", where X is either 85 or more (`Trustworthiness=1`, or "Trustworthy"), or 30 or less (`Trustworthiness=0`, or "Untrustworthy"). The rationale is that if a large fraction of inputs contributed to a value, this value is relatively trustworthy and can be used to decide between product A and B while if the fraction of inputs used to compute the average is small, then there is a large degree of uncertainty about the correct rating and it cannot be trusted in decision making.

We divided participants into 3 groups: For group 1, all values (including the designated highlight) were presented to participants without decoration. For group 2, the designated highlight was replaced with a blank cell. For group 3, the designated highlight was underlined and presented using one of seven formats for each table: red, blue, or green text; red, blue, or green backgrounds; or an appended asterisk (`Highlight format=0..6`). Furthermore, participants in group 3 were informed that clicking on the value would reveal additional information explaining the highlight — the number of missing values.

Research Questions: Our goal is threefold: (i) Help users to avoid mistakes based on hidden missing data, (ii) Avoid biasing users away from valid decisions, and (iii) Add minimal overheads when additional information is unnecessary. This leads us to 4 research questions and associated hypotheses.

RQ1- *In Relevant/Trustworthy trials, does group 3 act more like group 1 than group 2?* This question tests whether users make the right decision to trust the highlighted value when the uncertainty is unlikely to affect the result. We hypothesize that presenting users with information indicating that a value is Trustworthy will counteract suspicions raised by the highlight.

RQ2- *In Relevant/Untrustworthy trials, does group 3 act more like group 2 than group 1?* This question tests whether users correctly distrust highlighted values if their detail view indicates that they should not be trusted. We hypothesize that presenting users with information indicating that a value is Untrustworthy will counteract the presence of this invalid value.

RQ3- *Do participants in group 3 request the detail view more often in Relevant trials than Irrelevant trials?* This question tests whether users ignore highlights for values that are not relevant for their decision making process. That is, we are not pushing users to spend more time on irrelevant values by highlighting them. We hypothesize that users will want details to make a decision based on Relevant values, but not on Irrelevant ones;

RQ4- *Do participants in group 3 reach a decision faster when the highlighted value is Irrelevant than when it is Relevant?* The rationale behind this question is (i) to further investigate the questions behind RQ3 and (ii) to test whether users investigate the uncertainty relevant to their decision. We hypothesize that acquiring information will cause the user to spend more time deciding.

3.2 Results

The average total time taken by participant in groups 1 and 3 was 18 minutes for the entire study, while participants in group 2 took an average of 8 minutes to complete the study.

RQ1 [Partly Supported]: In answering this research question, we focus exclusively on the 28 Uncertain/Trustworthy trials. A chi-square test of independence examined the relation between correct choices in groups 3 and 1, and groups 3 and 2 respectively. In both tests, the null hypothesis is that the participant's group does not correlate with the rate of correct product selections.

Error context text should be revealed in order to determine trustworthiness of a value. Hence, we limited our analysis to participants who clicked at least once. The comparison between group 3 and group 1 was significant at α=0.05 with sample size 300 and degree of freedom 1 ($\chi^2(1, (N = 100) = 34.0767, p < 0.0001)$), and the comparison between group 3 and group 2 was also significant at α=0.05 with sample size 300 and degree of freedom 1 ($\chi^2(1, (N = 100) = 303.748, p < 0.0001)$). While there is a significant difference between group 3 and both other groups, the magnitude of the difference is smaller between groups 3 and 1.

RQ2 [Partly Supported]: We answer this research question through methodology similar to RQ1, but focusing instead on the 28 trials with Uncertain/Untrustworthy values. As before, a chi-square test of independence examined the relation between correct choices in groups 3 and 1, and groups 3 and 2 respectively. **H2** would be supported if the null hypothesis was not disproven for groups 2 and 3, and disproven for groups 1 and 3. Similar to RQ1 we limited our analysis to participants who clicked at least once.

The comparison between group 3 and group 1 was significant at α=0.05 with sample size 300 and degree of freedom 1 ($\chi^2(1, (N = 105) = 111.281, p < 0.0001)$), and the comparison between group 3 and group 2 was also significant at α=0.05 with sample size 300 and degree of freedom 1 ($\chi^2(1, (N = 105) = 13.444, p = 0.0002)$). While there is a significant difference between group 3 and both other groups, the magnitude of the difference is smaller between groups 3 and 2.

RQ3 [Supported]: **H3** states that users are more likely to click in cases where the Uncertain value is Relevant compared to cases where the value is Irrelevant. Only group 3 trials are relevant to this question. Because the data was not normally distributed and was not symmetrical, a sign test was run on results from the 300 group 3 participants. The null hypothesis is that the click rate is the same for both Relevant and Irrelevant trials. **H3** would be supported if there is a statistically significant chance of a non-zero median in the difference in the number of clicks between these two groups. The sign test with a confidence interval of 0.95 and a median of 0 indicates a significant result (N = 300, p = 0.034), supporting **H3**.

RQ4 [Supported]: To answer this research question, we focus exclusively on results for group 3. We excluded one outlier participant who took 84 seconds to complete all irrelevant trials; The remaining users completed all irrelevant trials within 46 seconds. To account for possible differences in difficulty between trials, the time measure for each participant on each trial was normalized by dividing it by the median time taken on that trial by group 1 participants. A sign test was applied to the normalized difference between total time that a group 3 participant spent on Relevant trials minus the time they spent on Irrelevant trials. The null hypothesis is that the median time difference is 0. The sign test showed a significant deviation from a 0 median in the positive direction at a confidence interval of 0.95 (N = 299, p = 0.015), supporting **H4**.

As a sanity check, we performed similar analysis with group 1. The data was normalized by dividing it by the median time taken on that trial. We would expect no significant deviation. A sign test performed on the normalized difference between total time that a group 1 participant spent on relevant trials minus the time they spent on Irrelevant trials did not show a significant deviation from a median of 0 at a confidence interval of 0.95 (N = 300, p = 0.073).

3.3 Qualitative feedback

Out of 300 participants in group 3, a total of 113 clicked to get additional information in one or more trials. 92 users clicked in both relevant and not relevant datasets. 8 clicked only in relevant datasets and 13 clicked only in not relevant datasets.

We observed a slight uptick in very high confidence levels (Strongly Agree) for Not Relevant over Relevant datasets in group 3, but a similar uptick appears in group 1 — a chi-square test shows no significant difference between group 3 and group 1 rates ($\chi^2(1, (N = 300) = 2.540706, p = 0.1109)$). However, the distribution of confidence exhibits a significant shift towards decreased confidence for group 3 relative to group 1 ($\chi^2(1, (N = 300) = 506.580, p < 0.0001)$) – fewer Strongly Agree responses, and more Strongly Disagree, Disagree, or Neither Agree nor Disagree responses.

4 RELATED WORK

Trust and Context: Research carried out in several domains including healthcare, weather prediction, transportation, and more, indicates that displaying context, particularly when related to uncertainty in the data, can improve trust and decision-making [3, 15, 16, 22]. When such information is absent, users take unnecessary precautions [15] as trust in the data decreases.

Emphasis in Non-Tabular Data: Emphasis in data plots has been explored extensively, particularly in the context of missing data or imputed values [4, 11, 24]. Leveraging visual saliency helps draw analyst attention to imputed or missing values. Notably, point emphasis becomes more effective when annotations are paired with contextual details about why values are missing [6].

Putting Data In Context: Many data exploration systems put data in context by displaying task-specific summary statistics and visualizations. For example Wrangler [17] uses a "completeness bar" to emphasize missing or invalid values, and histograms to help identify outliers. Although such visualizations can be helpful, they are often (i) task specific and (ii) generated purely by data profiling [1]. DataSense (i) targets a more general use case, and (ii) also permits the use of provenance to generate context.

Annotation Management: A key source of context is freeform prose documentation. For this, DataSense relies on a class of techniques called annotation management [7, 13], where freeform prose annotations are propagated through queries. Annotation management poses a range of challenges including representation, archival, and propagation [13]; and summarization [26].

Explanations: Another class of automated documentation is query explanations. Modern approaches include explaining the outcome of a query results using summarizations [23]; errors and outliers diagnosis [25]; preprocessing database for explanations [20]; and identifying important inputs [15].

5 CONCLUSIONS AND OPEN CHALLENGES

We have proposed an abstraction, display-agnostic data documentation, to link data and documentation by decoupling the techniques used to generate the documentation for the modality in which the data is displayed. We presented presented DataSense, a concrete implementation of that abstraction over Apache Spark. Through a user study, we showed that display-agnostic data documentation style abstractions can help users to make better decisions, while staying unobtrusive when the documentation is irrelevant.

More work is needed to realize our vision of a system that integrates data display and documentation. First, DataSense relies on a set of automatic documentation extractors, analogous to similar systems in IDEs. However, in contrast to programs which are written by humans and, thus, are moderate in size, the size of most datasets requires a more scalable and flexible way to associate documentation with data. We envision DAD allowing users to declaratively document dataset elements. A key part of this process is improving the process for inferring documentation from source data. In contrast to existing work on annotation propagation where propagation rules are hard-coded [5] or defined by the query author [7, 9], we want to allow annotation creators to control propagation semantics or allow annotations to be combined [18]. In principle, these are analogous problems, but unlike query authors, annotation authors have no control over how the data will be queried and must prepare rules that cover all possible queries.

Another challenge is redundant computation across documentation modules. DataSense does allow limited work sharing by allowing documentation modules to instrument the user's dataframe. However, this is not always appropriate. For example, computing mean and standard deviation (for outlier highlighting) can share work with computing the first 20 rows of a dataset (for Spark's `show()` method), but not as part of the same query. We also observe that a generic mechanism for instrumenting queries can help documentation modules to avoid interfering with each-other's instrumentation or with the data being displayed.

REFERENCES

[1] Ziawasch Abedjan. 2019. Data Profiling. In *Encyclopedia of Big Data Technologies.*

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

[3] Stavros Antifakos, Adrian Schwaninger, and Bernt Schiele. 2004. Evaluating the Effects of Displaying Uncertainty in Context-Aware Applications. In *UbiComp*. 54–69.

[4] Christian Bors, Markus Bögl, Theresia Gschwandtner, and Silvia Miksch. 2017. Visual support for rastering of unequally spaced time series. In *VINCI*.

[5] Michael Brachmann, William Spoth, Oliver Kennedy, Boris Glavic, Heiko Mueller, Sonia Castelo, Carlos Bautista, and Juliana Freire. 2020. Your notebook is not crumby enough, REPLace it. In *CIDR*.

[6] Paolo Buono, Aleks Aris, Catherine Plaisant, Amir Khella, and Ben Shneiderman. 2005. Interactive pattern search in time series. In *VADA*, Vol. 5669. 175–186.

[7] Laura Chiticariu, Wang-Chiew Tan, and Gaurav Vijayvargiya. 2005. DBNotes: a post-it system for relational databases based on provenance. In *SIGMOD*. 942–944.

[8] New York City. 2008-2016. NYC OpenData. https://nycopendata.socrata.com/.

[9] Mohamed Y Eltabakh, Walid G Aref, Ahmed K Elmagarmid, Mourad Ouzzani, and Yasin N Silva. 2009. Supporting annotations on relations. In *EDBT*. 379–390.

[10] Su Feng, Aaron Huber, Boris Glavic, and Oliver Kennedy. 2019. Uncertainty Annotated Databases-A Lightweight Approach for Approximating Certain Answers. In *SIGMOD*. 1313–1330.

[11] Sara Johansson Fernstad and Robert C Glen. 2014. Visual analysis of missing data—To see what isn't there. In *VAST*.

[12] Juliana Freire, Aline Bessa, Fernando Chirigati, Huy T. Vo, and Kai Zhao. 2016. Exploring What not to Clean in Urban Data: A Study Using New York City Taxi Trips. *IEEE Data Eng. Bull.* 39, 2 (2016), 63–77.

[13] Floris Geerts, Anastasios Kementsietsidis, and Diego Milano. 2006. MONDRIAN: Annotating and Querying Databases through Colors and Blocks. In *ICDE*.

[14] Madelon Hulsebos, Kevin Zeng Hu, Michiel A. Bakker, Emanuel Zgraggen, Arvind Satyanarayan, Tim Kraska, Çagatay Demiralp, and César A. Hidalgo. 2019. Sherlock: A Deep Learning Approach to Semantic Data Type Detection. In *KDD*.

[15] Susan Joslyn and Jared LeClerc. 2013. Decisions with uncertainty: the glass half full. *CDP* 22, 4 (2013), 308–315.

[16] Malte F Jung, David Sirkin, Turgut M Gür, and Martin Steinert. 2015. Displayed uncertainty improves driving experience and behavior: The case of range anxiety in an electric car. In *CHI*.

[17] Sean Kandel, Andreas Paepcke, Joseph M. Hellerstein, and Jeffrey Heer. 2011. Wrangler: interactive visual specification of data transformation scripts. In *CHI*.

[18] Egor V Kostylev and Peter Buneman. 2012. Combining dependent annotations for relational algebra. In *ICDT*. 196–207.

[19] Seokki Lee, Bertram Ludäscher, and Boris Glavic. 2020. Approximate Summaries for Why and Why-not Provenance. *pVLDB* 13, 6 (2020), 912–924.

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

[21] William Spoth, Poonam Kumari, Oliver Kennedy, and Fatemeh Nargesian. 2020. Loki: Streamlining Integration and Enrichment. In *HILDA*.

[22] Antony Unwin, George Hawkins, Heike Hofmann, and Bernd Siegl. 1996. Interactive graphics for data sets with missing values—MANET. *JCGS* 5, 2 (1996), 113–122.

[23] Michael Vollmer, Lukasz Golab, Klemens Böhm, and Divesh Srivastava. 2019. Informative Summarization of Numeric Data. In *SSDBM*.

[24] BL William Wong and Margaret Varga. 2012. Black holes, keyholes and brown worms: Challenges in sense making. In *HFES*, Vol. 56. 287–291.

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

[26] Dongqing Xiao and Mohamed Y Eltabakh. 2014. InsightNotes: summary-based annotation management in relational databases. In *SIGMOD*. 661–672.