RuleScope: Semantic-aware Authoring of Data Validation Rules

Zhongsu Luo, Di Weng, Jiawen Zhu, Shuhan Liu, Xiwen Cai, Ran Chen, Kai Xiong, Jiajun Zhu, Xinhuan Shu, Yingcai Wu, *Senior Member, IEEE*

***Abstract*—Data validation is a crucial step in data analytics workflows that assesses and ensures the reliability of data flowing into analytical processes. One common approach to data validation involves defining validation rules, which provide explicit constraints and conditions that data must satisfy. However, creating accurate and effective validation rules remains challenging for many practitioners. This challenge stems from the need for practitioners to understand both data structures and their domain-specific semantic relationships. Recent studies have proposed automated approaches to generate validation rules by deriving patterns from data properties. However, these approaches generate rules with limited interpretability and lack support for rule verification and modification, making the rules difficult to understand and adapt. To address these limitations in current validation rule authoring approaches, we present RuleScope, an interactive system for authoring data validation rules through semantic-aware rule generation, visualization, and refinement. RuleScope employs an LLM-based workflow to generate interpretable rules by analyzing data semantics and incorporating domain knowledge. To facilitate rule comprehension, we design a matrix-based visualization that helps users understand rules and analyze validation results. Additionally, RuleScope enables users to interactively refine rules. We evaluate the LLM-based workflow through model evaluation on datasets from different domains and assess RuleScope's usability and effectiveness through two case studies and a user study.**

***Index Terms*—data validation, data quality, visualization, interactive tool.**

I. Introduction

Increased data accessibility allows organizations to derive significant value from its analysis, enabling informed decision-making and improved strategic planning [1], [2]. However, the accuracy of data analysis and the reliability of subsequent decisions can be compromised by data errors [1], [3], potentially leading to considerable financial consequences [4]–[7].

Data validation serves as a critical step for improving data quality in data analytics workflows, aiming to identify errors and ensure the correctness and validity of input data. A common data validation approach is to detect erroneous data by defining validation rules. For example, a typical validation rule that checks whether a high-risk blood pressure dataset is correct might be expressed as: `blood_pressure > 140 mmHg AND age > 60`. These validation rules serve to encapsulate knowledge about the semantic correctness of data, thereby facilitating the effective monitoring and cleaning of subsequently acquired data.

However, manually authoring these rules remains challenging for practitioners. First, effective rule creation requires analyzing data characteristics combined with domain knowledge, while working with massive, complex datasets remains difficult and time-consuming. Second, practitioners must translate conceptual rules into executable implementations by defining detection functions and parameters. Furthermore, this process typically requires iterative refinement based on validation results. Recent studies have employed machine learning approaches [8]–[11] to generate validation rules from data based on statistical patterns. However, the generated rules typically lack interpretable semantic meaning that aligns with domain knowledge, thereby impeding human understanding and the ability to refine and optimize these rules effectively. Moreover, existing approaches focus on rule generation while neglecting subsequent verification and refinement. This oversight necessitates intuitive visualizations to assist users in interpreting diverse rules and their effects on data, while supporting iterative refinement through interactive exploration and modification.

Motivated by the limitations of prior studies, we collaborated with four domain experts to design RuleScope, an interactive system that supports the generation, verification, and refinement of data validation rules based on data semantics. Developing such a system poses three research challenges:

Generating interpretable validation rules based on data semantics. The first challenge lies in improving the interpretability of the generated rules by aligning rules and data semantics. The current approaches mainly rely on profiling statistical characteristics and fitting data distributions, resulting in complicated rules like "$0.67 \times \text{blood\_pressure} + 1.5 \times \text{age} > 183.8$" for the high-risk blood pressure scenario. The interpretability of these rules is crucial in facilitating subsequent human-in-the-loop rule verification and refinement.

Verifying complex validation rules based on their results. Users need to verify both validation rules and their results to ensure rule effectiveness. However, this verification process is complicated by diverse rule types and complex rules involving data relations and conditional dependencies

Z. Luo, S. Liu, R. Chen, J. Zhu and Y. Wu are with State Key Lab of CAD&CG, Zhejiang University, Hangzhou, China. E-mail: {zhongsuluo, shliu, chenran928, jiajunzhuchris, ycwu}@zju.edu.cn.

D. Weng and J. Zhu are with School of Software Technology, Zhejiang University, Ningbo, China. E-mail: {dweng, jiawenzhu}@zju.edu.cn. D. Weng is the corresponding author.

X. Cai is with Department of Digital Intelligence, China Mobile, Shenzhen, China. E-mail: caixiwen@chinamobile.com

K. Xiong is with HiThink Research, Hangzhou, Zhejiang, China E-mail: xiongkai@myhexin.com

X. Shu is with School of Computing, Newcastle University, Newcastle Upon Tyne, United Kingdom. E-mail: xinhuan.shu@newcastle.ac.uk

The source code of RuleScope is available under the MIT License at https://github.com/rulescopevis/RuleScope.

Manuscript received xxx. xx, 2025; revised xxx. xx, 2025.

(e.g., If blood_pressure $>$ 140 AND age $>$ 60 THEN high risk = TRUE), requiring users to examine relationships across multiple data elements. It remains a significant challenge to effectively support users in the verification process of these complex rules and their results.

Refining validation rules to balance rule strictness and accuracy. Data validation authoring requires iterative refinement of rules through observation of their effects on datasets. However, intuitively translating the users' assessment on the quality of validation rules into data boundary refinement of these rules remains a challenging task. This refinement process must balance between the strict rules that may reject valid data and the inaccurate rules that may miss erroneous data.

To address these challenges, we established a validation rule classification through expert interviews and literature review [1], [8], [12]–[15]. Following this classification, we constructed a rule generation workflow using locally-deployed large language models (LLMs) that generates interpretable validation rules by combining domain knowledge and semantic analysis. Drawing inspiration from MatrixWave [16], we designed a matrix-based visualization to help users understand rules and analyze validation results. We also implemented various interactions, including natural language interface, example specification and parameter fine-tuning, to support users in refining validation rules flexibly. To evaluate RuleScope, we analyzed the performance of the rule generation workflow across different domains and evaluated the system through two case studies and a user study. The results demonstrate how RuleScope improves efficiency in the validation process.

Our main contributions are summarized as follows:

- We propose an LLM-based workflow for generating interpretable validation rules through semantic analysis and domain knowledge;
- We design a matrix-based visualization for facilitating the interpretation of validation rules and their results;
- We implement RuleScope, an interactive tool for authoring data validation rules that supports semantic-aware rule generation, verification, and refinement.

II. Related Work

A. Data Validation

Data validation is crucial for ensuring data quality in data analysis workflows. Traditionally, users write validation rules using programming languages such as Python or SQL. However, these general-purpose languages lack specialized features for data validation, which often leads to complex implementations and higher development costs. In response to these challenges, several companies have developed specialized tools such as Google's TensorFlow Data Validation (TFDV) [17], [18], Amazon's Deequ [19], [20] and LinkedIn's Data Sentinel [21], [22]. These tools use declarative approaches for high-level rule specification. Nevertheless, these tools still necessitate manual rule definition, and it is difficult for users to define appropriate validation rules [11].

Error detection algorithm. Error detection algorithms are crucial due to the time-consuming and labor-intensive nature of manually defining validation rules. Tools such as Microsoft Excel [23], [24] offer nine predefined validation rules. Trifacta [25], Power BI [26], and Talend [27] support pattern anomaly detection for predefined data types, such as email addresses. OpenRefine [28] employs fuzzy-group-by methods to identify spelling errors. Error detection is a well-established research area. Feature-based methods detect outliers by analyzing statistical properties, primarily in time series data where patterns help identify anomalies [29]–[33]. Constraint methods detect data anomalies by setting conditions to identify errors, often validating data across multiple columns [34]–[38], such as functional dependencies. Semi-automated methods [12], [39]–[41] have been devised to validate data. Other approaches automatically detect errors and generate validation rules based on profiling statistical characteristics and fitting data distributions [9], [13], [33]. Tu et al. [8] analyze past executions of data pipelines to automatically generate validation rules. Picket [42] performs self-supervised data validation for machine learning pipelines. Despite these advancements, existing algorithms for data validation have limitations. Some require manual rule-setting, while automated algorithms often lack flexibility and depend on clean datasets for rule generation, failing to meet user needs. Thus, user involvement remains essential in the data validation process [43].

Visual and interactive validation. Visualization and interaction techniques assist users in participating in the data validation process. Data visualization enables users to discover latent insights and enhance cognitive abilities [44]. Numerous data management systems [21], [25], [26], [28] and data profiling tools [43], [45] use visualization to help users understand data distributions, identify trends, and detect anomalies. Early approaches primarily use basic charts to inspect univariate distributions, such as bar charts [46]–[48], line charts [49], and area charts [45], [50]. For instance, Wrangler [48] employs interactive bar charts as data quality meters to visualize the distribution of valid, invalid, and missing values based on inferred semantic roles. Although these conventional visualizations are easy to interpret, they have limitations in representing complex data characteristics and relationships. To examine attribute relationships, researchers extend these techniques with bivariate and multivariate visualizations. Scatterplots are commonly used [43], [51]–[53] to reveal bivariate patterns. Profiler [43] extends this approach with binned scatterplots and scatterplot matrices (SPLOMs) to visualize correlations and highlight multivariate outliers through linked interactions. However, these techniques primarily depict statistical patterns rather than encoding validation logic. For data streams, multidimensional data, and sequential data, prior work has employed visualizations such as node-link graphs [54]–[56], parallel coordinates plots [57]–[59], and sequence matrices [16], [60] to represent data relationships. However, reliance on visual inspection alone presents significant cognitive challenges when analyzing large-scale datasets or complex relationships. Recent work has further explored interactive visual analytics for diagnosing and refining data quality issues in machine learning workflows [61]–[64]. For example, LabelInspect [61] supports the validation of uncertain labels and unreliable workers through coordinated visualizations and iterative feedback propagation. Related task-specific systems have also been developed for

interactive labeling and annotation correction in specialized domains, such as suspicious cluster labeling and action annotation correction [65], [66]. However, these approaches target specific machine learning tasks rather than explicit validation rule generation, verification, and refinement for tabular data.

To address these limitations, we designed RuleScope, an interactive system supporting the generation, verification, and refinement of data validation rules. RuleScope leverages LLMs to analyze data semantics and domain knowledge for rule generation, employs a matrix-based visualization for rule verification, and enables interactive rule refinement.

B. LLMs for Exploratory Data Analysis

Large language models (LLMs) have demonstrated significant capabilities, with models such as GPT-5.2 [67] and Claude-4.5 [68]. Researchers have successfully deployed these models across numerous tasks, including visualization recommendation [69], evaluation [70], color palette design [71], web design [72], and data transformation [73]. Building on their comprehensive domain knowledge and sophisticated reasoning capabilities, researchers have increasingly applied these models to exploratory data analysis (EDA) tasks.

Current research leveraging LLMs for EDA can be categorized into two primary approaches. The first approach improves efficiency during data exploration. Data Formulator [73] employs LLMs to interpret user intent and facilitate data transformation operations, whereas chat2vis [74] enables visualization generation through natural language inputs. Some studies [69], [75] decompose this process into multiple sequential steps, incrementally generating visualizations based on both data characteristics and natural language queries. The second approach emphasizes enhancing users' analytical capabilities by discovering data insights. AI Threads [76] implements progressive data analysis through multi-turn conversations, and InsightPilot [77] utilizes LLMs as an insight engine to facilitate insight discovery. However, these approaches predominantly focus on identifying analytical features that reflect knowledge and facts within the data, while lacking sufficient emphasis on data quality analysis and exploration.

Inspired by these studies, we observe that LLMs' ability to help users uncover data insights can be extended to systematically identify and address data quality issues. Unlike previous approaches focused on analytical features, our research specifically targets the challenges in data validation through semantic understanding. Based on this observation, we propose an LLM-based workflow that constructs interpretable validation rules by analyzing data semantics and leveraging domain knowledge to facilitate validation rule authoring.

III. Classification of Rules and Task Abstraction

This section presents the background of our study, the classification of validation rules, and the user requirements.

A. Background

Over the past year, we collaborated with four domain experts (E1-4) to understand how they author validation rules for data validation tasks. E1 and E2 are industry practitioners from the business analytics and energy engineering sectors, respectively, each with over three years of experience in domain-specific data cleaning and analysis. Both of them conduct data validation tasks on production data as part of their daily responsibilities. E3 is a senior researcher in urban computing with experience in urban data analysis and visualization, while E4 is a Ph.D. candidate conducting research in sports science. Both E3 and E4 work with datasets from various sources. Data validation and error detection constitute critical components of their workflow, directly impacting the accuracy of their visual analytics and model training processes.

To understand data validation processes and challenges in practice, we interviewed experts about their validation experiences. During the interviews, experts described their recently used datasets, validation procedures, and implemented rules. Our investigation revealed that experts implement validation rules either programmatically (using Python or Amazon Deequ) or through interactive tools (like Excel or Data Sentinel). Rule authoring follows an iterative process where experts formulate initial rules based on domain knowledge and data semantics, then refine them after examining validation results. This refinement is necessary as initial rules rarely achieve optimal performance without adjustments. Such a data validation workflow presents three major challenges: (1) formulating validation rules requires significant effort to understand data semantics, examine data characteristics, and incorporate domain knowledge; (2) interpreting validation results demands experts to comprehend rules and assess their appropriateness based on validation results; and (3) refining rules necessitates context switching between validation results and rule implementation. Based on these findings, the goal of this study is to develop a visual analytics solution that addresses these challenges by providing integrated tools for rule formulation, result interpretation, and iterative refinement within a unified interface that supports data practitioners in creating effective validation rules. To achieve this goal, our research focuses on two fundamental questions: *what to validate* and *how to validate*. The former concerns identifying which data elements, relationships, and properties should be subject to validation. The latter addresses how we can effectively support data practitioners in authoring validation rules.

B. Classification of Validation Rules

To understand *what to validate*, we first established a classification of validation rules by analyzing domain experts' validation practices, identifying two primary categories:

- **Cell-level validation rules** check individual data values independently, examining properties such as data types, value ranges, and format constraints. These rules are self-contained as they can be validated without referencing any other data cells.
- **Dependency validation rules** define relationships between multiple data cells. For example, these rules may comprise simple comparisons (e.g., a cell's value must be less than another) or complex conditional constraints (e.g., when a cell contains a specific value, another cell

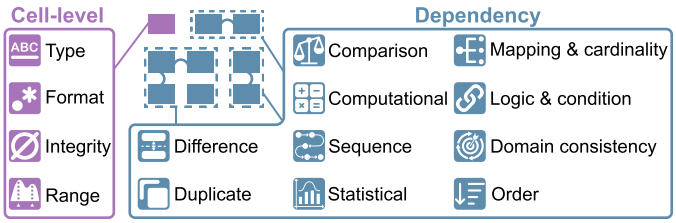

Fig. 1. The classification of validation rules consists of 2 categories, Cell-level and Dependency, which are further divided into 14 subcategories.

must satisfy a particular condition). Due to their intrinsic complexity in generation, verification, and refinement, dependency rules constitute our primary focus.

To further refine this initial categorization, we reviewed literature on error detection [12], [13], [78], data quality [1], [14], [15], [79] and data validation [8], [9], [80]. Building upon this literature review and continued bi-weekly discussions with the domain experts, we further divided our initial categorization of validation rules into 14 detailed subcategories, as illustrated in fig. 1. These subcategories are described as follows:

- **Type** ensures data conforms to specified types, including character, numeric, and date.
- **Format** verifies adherence to specific format requirements, such as applying regular expressions to enforce character length and pattern, or validating date-time formats (e.g., yyyy-mm-dd).
- **Integrity** checks for missing or anomalous data, encompassing both standard missing values (null values, np.nan) and domain-specific indicators (such as using -1 to denote missing value).
- **Range** confirms data values fall within defined boundaries, such as ensuring PH values remain within the range of 0 to 14.
- **Comparison relations** examines comparative relationships between values. For numeric or date data, this involves magnitude comparisons, such as PM2.5 $\leq$ PM10 in atmospheric monitoring data. For character data, this involves comparing string lengths or substring relationships, such as the relationship between email addresses and email domains, where the email domain must be a substring of the email address.
- **Computational relations** ensure relationships between columns through direct calculations (e.g., a student's total score equals the sum of individual subject scores).
- **Logical and condition** checks conditional relationships between columns, where certain values act as conditions that impose constraints on other values. When specific conditions are met in one column, they trigger validation rules for related columns. For instance, when age is less than 18, minor_status must be True.
- **Mapping and cardinality** validates predefined mappings between columns. Unlike logical and condition, which activates a constraint only when a triggering condition is met in another column, this rule defines a static structural association where each value must conform to a reference set. It covers value-based dependencies where one value determines another (e.g., postal codes must match their designated geographic regions), as well as cardinality constraints such as one-to-many relationships (e.g., one department maps to multiple courses).
- **Domain consistency** validates the semantic appropriateness of column values. It checks that values belong to the correct category (e.g., no country names in a city column) and that the same entity is represented consistently (e.g., "New York" rather than "NY"). As a semantic-level validation, it concerns the meaning of values within their column context, whereas range validates numerical boundaries and format enforces syntactic patterns.
- **Statistical** examines numerical values in a column against statistical rules, such as outlier detection and distribution patterns, where age values should follow expected population distribution.
- **Difference** evaluates adjacent data points, ranging from threshold checking (e.g., temperature changes must not exceed 5°C, or price fluctuations must stay within 15%) to distances across multiple dimensions (e.g., in a 3D space, successive points (x_1, y_1, z_1) and (x_2, y_2, z_2) have reasonable Euclidean distances).
- **Duplicate** examines value uniqueness, ranging from individual columns (e.g., employee ID numbers as primary keys) to multiple column combinations (e.g., in atmospheric monitoring data, while location and timestamp may have duplicate values individually, their combination must be unique, as one location can only have one measurement at any given time).
- **Order** verifies that consecutive values in a column follow a predefined monotonic direction, such as record timestamps appearing in ascending order. In contrast, difference rules check the magnitude of change between adjacent values rather than its direction, ensuring it stays within an acceptable threshold.
- **Sequence** examines transitions between data following sequential relations, such as in transaction records where "order placement" should be followed by "payment".

It is important to note that this classification deliberately emphasizes **dependency validation rules** over **cell-level** ones, reflecting the specific needs expressed by our domain experts during discussions. Specifically, our taxonomy prioritizes rule semantics and validation purposes over structural disjointness, enabling intuitive rule authoring aligned with experts' mental models. While this taxonomy satisfies the current requirements of our collaborators and provides a structured framework for the interactive rule construction method, we acknowledge its potential limitations in scope and comprehensiveness. Such a taxonomy should be viewed as an evolving framework rather than a definitive categorization. We encourage future research to expand, refine, or restructure this taxonomy as additional validation scenarios emerge or as the method is applied across different domains with unique validation requirements.

C. Requirement Analysis

To develop an effective interactive system that supports the authoring of these validation rules and address the question *how to validate*, we conducted weekly meetings with four

domain experts over three months. We designed a three-phase process to systematically elicit requirements covering the complete rule authoring workflow. In Phase 1 (Rule Initialization, 5 weeks), we discussed challenges in manual rule construction and explored what semantic information generated rules should contain. Through iterative sketching of example rules, we clarified requirements for automating rule generation while ensuring interpretability. In Phase 2 (Rule Comprehension and Analysis, 6 weeks), we discussed challenges in understanding rule definitions and analyzing validation results, and explored how to visually present them. Iterative prototyping of alternative visualizations helped clarify requirements for revealing rule logic and supporting result analysis. In Phase 3 (Rule Refinement, 3 weeks), we discussed challenges in iterative refinement and context switching between modification and evaluation. Collaborative exploration and sketching clarified requirements for different rule types while minimizing context switching. Through these three phases, we identified four key user requirements that guided RuleScope's design as follows.

R1: Generate semantic-aware validation rules. Users need to construct validation rules based on data semantics and domain knowledge, ensuring these rules are semantic-aware and aligned with domain expertise. Semantic-aware rules capture meaningful data relationships and reflect real-world constraints, supporting both accurate validation and iterative refinement. However, manually constructing these rules is time-consuming and challenging. As E2 emphasized: *"Despite having sufficient domain knowledge, I still find it challenging to initialize validation rules."* Therefore, the system should generate semantic-aware validation rules that align with domain expertise, reducing users' burden in the initial phase.

R2: Reveal validation rule definitions. Users need to understand both the definitions and types of validation rules. E4 mentioned: *"Before applying validation rules, I need to verify if these rules actually make sense for my data and business context, but this verification process is time-consuming and labor-intensive."* The system should provide intuitive visualizations of validation rules, helping users comprehend and evaluate rules.

R3: Enable data-driven validation result analysis. After applying rules, users need to analyze how these rules perform against actual data to determine their appropriateness. As E3 noted, *"It would be helpful to display which validation rules are being violated/satisfied by the current data."* The system should enable users to explore compliant and violating data instances, revealing whether rules need adjustment for being too strict or too lenient.

R4: Support flexible rule refinement. Users need to iteratively refine rules by making adjustments and then examining how these changes affect the validation outcomes, requiring frequent shifts in focus between rule editing and result assessment. Furthermore, preferred modification approaches vary depending on the rule types. For simple rules, users prefer direct parameter modification, while for complex rules such as dependencies, they require natural language or other intuitive methods to express their intentions. Therefore, the system should support flexible rule refinement while minimizing context switching.

IV. RuleScope Workflow

This section presents RuleScope's workflow for validation rule generation, implementation, and refinement. Our workflow addresses the key challenge of generating interpretable validation rules by leveraging domain knowledge, semantics, and characteristics of datasets. We achieve this through three technical innovations: (1) a taxonomy-driven design that enables generating validation rules in a structured format, (2) data preprocessing procedures that prepare raw data into LLM-comprehensible formats tailored to different rule types, and (3) a prompting strategy that guides LLMs to extract data semantics effectively. The workflow consists of three components: an LLM-based validation rule recommender (section IV-A), a rule implementer (section IV-B), and a rule refiner (section IV-C).

A. Validation Rule Recommender

The Validation Rule Recommender (fig. 2A) generates validation rules (R1). Based on expert interviews, we identified that the generation of validation rules should integrate both domain knowledge and data semantics. The recommendation should consider two key rule types: cell-level rules and dependency rules. We adopted LLMs as our solution approach because they effectively encapsulate domain knowledge [81] and possess the ability to understand data semantics [82]. Several studies have applied LLMs to exploratory data analysis (EDA) [73], [74], [76], [77] and data management tasks [83], [84]. However, applying LLMs to validation rule generation presents two main challenges: effectively representing complex tabular data and handling diverse rule types with distinct reasoning requirements. To address both challenges, we optimized tabular data representation and implemented a type-specific rule generation strategy. The Recommender consists of a two-stage pipeline to generate rules:

Tabular data preprocessing. To generate reliable validation rules, LLMs require comprehensive tabular data characteristics. However, feeding complete data tables to LLMs is infeasible due to token limitations and efficiency concerns, potentially degrading rule generation performance. Our preprocessing approach (fig. 2A1) categorizes columns by data types for subsequent processing. The preprocessing steps include: (1) data compression: merging identical values with their frequencies and proportions, reducing input size for categorical data, repetitive values, and datasets with high null value density; (2) statistical computation: extracting key metrics from numeric data (mean, median, outlier detection); (3) pattern extraction: analyzing various data patterns, e.g., sequential relationships between consecutive values, to facilitate inference of underlying data dependencies; (4) domain detection: analyzing column names and data values semantically to identify the specific data domain, which enables the generation of contextually relevant dataset descriptions that enhance the precision of validation rule recommendations. Detailed preprocessing procedures are described in supplementary materials Sec. 2.1. The extracted information is structured in JSON format for subsequent rule recommendation.

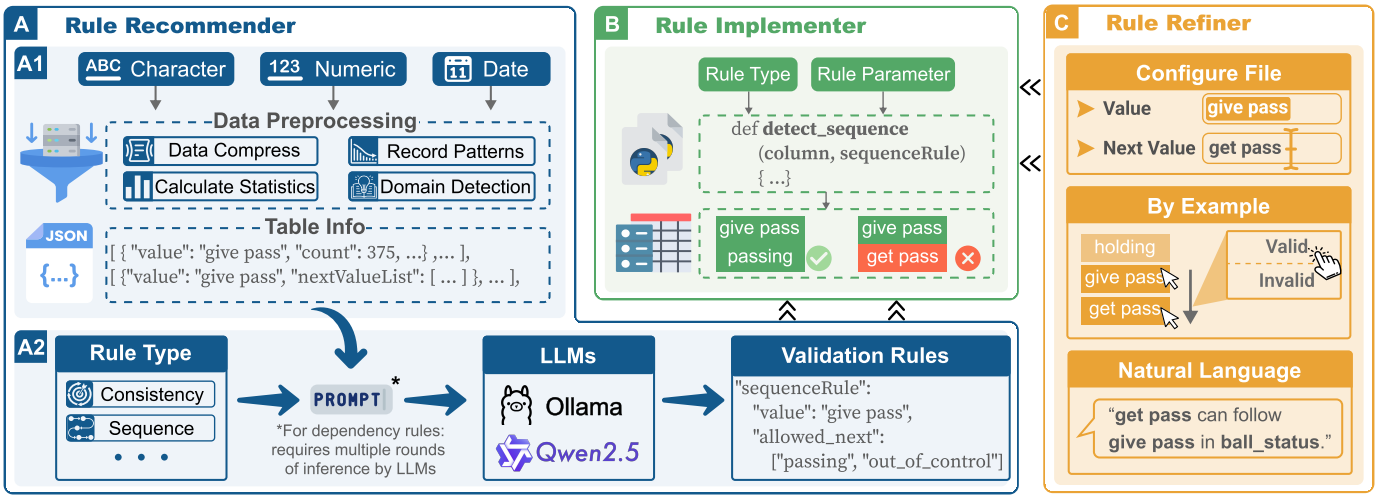

Fig. 2. RuleScope workflow. (A) Rule Recommender generates validation rules for the current table, including data preprocessing (A1) and LLM-based validation rule generation (A2); (B) Rule Implementer converts validation rules into executable functions; (C) Rule Refiner facilitates user refinements to rules through direct parameter modification, example-based demonstration, and natural language instruction.

LLM-based rule recommendation. The second stage implements a type-specific validation rule generation strategy (fig. 2A2), where rules are recommended sequentially by rule type. This sequential approach is necessary for two main reasons: (1) rule types have distinct data requirements, and including all table information in a single prompt would exceed the token limitations of LLMs; (2) concurrent generation of multiple rule types would overload the model's capacity to maintain logical consistency across different validation patterns. To address these challenges, we construct prompts for each type by extracting relevant information from the preprocessed data based on specific requirements. For dependency rules, which involve complex relationships requiring substantial computation, we adopt a semantic-guided approach instead of exhaustive pairwise analysis. This approach operates in two phases: (1) LLMs identify potentially related variables through semantic analysis of column names and examination of sample data; (2) these candidate relationships undergo detailed analysis to formulate precise rules. For example, when generating logical and condition rules, LLMs first identify potentially related columns, then analyze their data characteristics for rule inference. Ultimately, the Recommender generates different types of validation rules along with their parameters, which typically fall into two categories: (1) parameters that represent the constraints of the current validation rule; and (2) conditional parameters required before rule execution (such as filter, order, etc.). For instance, sequence-type validation rules require data to be sorted by specific columns prior to validation, thus including order condition parameters. These generated rules with their respective parameters are then consolidated into a JSON configuration file for the current dataset. This format enables our system to parse and render each rule as interpretable descriptions and visualizations, supports transpilation into executable scripts (e.g., Python), and simplifies maintenance as dataset requirements evolve. We implemented our Recommender using Qwen2.5 [85], an open-source model family ranging from 7B to 72B parameters. We selected appropriate model sizes based on task complexity, balancing inference accuracy and computational efficiency for each type of rule generation task (see supplemental material Sec. 3 for details). We chose local deployment through Ollama [86] for faster communication, enhanced control, and support for longer context length. We present the evaluation of our rule recommendation models in Section VI-A.

B. Validation Rule Implementer

The Validation Rule Implementer (fig. 2B) transforms the validation rule configuration file into executable validation functions. This configuration file is either generated by the Rule Recommender (section IV-A) or defined by users (section IV-C). For each rule, the Implementer identifies its type and invokes the corresponding template function, passing the rule's specific parameters as function arguments. These parameters describe both the constraints that data must satisfy under this validation rule and the conditions for data preprocessing before validation, such as filtering and sorting requirements. For instance, when implementing sequence rules, the parameters specify the sequence constraints the data should follow, as well as the ordering conditions that determine how the dataset should be sorted prior to validation. The template functions first apply these preprocessing transformations, then execute the validation logic against the preprocessed data. The Implementer ultimately returns the indices of non-compliant data entries, which are then used for subsequent visualization and analysis of validation results.

C. Validation Rule Refiner

The requirement analysis identified the need for flexible rule refinement (R4), which motivated the development of the Validation Rule Refiner (fig. 2C). Based on the refinement needs identified in our requirements analysis and iterative design with domain experts, we designed three interaction patterns to support different refinement scenarios: (1) Direct parameter

modification for simple refinement, such as True/False settings in missing value detection; (2) Example-driven refinement, where experts demonstrate valid/invalid cases by selecting data instances; (3) Natural language interaction, where experts use LLMs to adjust complex rule scripts through natural language. For direct parameter modifications, the Refiner supports editing the validation rule configuration file directly, while for example-driven and natural language refinements, it employs LLMs to interpret user inputs and generate appropriate rule modifications to update the configuration file. The Refiner subsequently triggers the Implementer (section IV-B) with modified parameters to re-validate the dataset.

Example-driven refinement uses selected data instances to modify validation rules, allowing users to refine rules through specific examples rather than abstract specifications. The Refiner identifies target rule types based on selection locations. For instance, it recognizes inter-column rules when selections span different columns within the same row. It then evaluates existing rules against the provided valid/invalid examples. For valid examples, it examines satisfying rules to determine if stricter conditions are needed, while analyzing non-conforming rules for possible modifications. For invalid examples, it follows similar steps with complementary criteria. Finally, the Refiner analyzes example characteristics to assess the need for rule additions or removals.

Natural language refinement enables intuitive rule modification. The Refiner uses LLMs to parse users' natural language descriptions, extracting validation targets and parameters. For example, in sports event data, a validation rule might specify that in the `ball_status` column, "get pass" cannot be followed by "give pass". In this rule, the column represents the target and the invalid sequence serves as the parameter. It then compares these extracted targets and parameters against existing rules, adding, modifying, or deleting rules as appropriate. We evaluated the accuracy of this natural language refinement. The evaluation dataset and results are detailed in Sec. 3.6 of the supplemental material.

V. RuleScope Interface

The visual interface of RuleScope comprises three views (fig. 3): (A) The Validate View presents rules and validation results (R2) and supports rule refinement through interaction with visualizations (R4); (B) the Data View presents tabular data with detailed validation results (R3), allowing rule refinement through table-based example selection (R4); (C) the Detail View displays an overview of currently applied validation rules (R2), offering parameter adjustment and natural language interfaces for refining validation rules (R4).

A. Validate View

The Validate View (fig. 3A) presents validation rules and their corresponding results. The upper-right corner contains an Upload File button for dataset uploading and an Export button. The Export button allows users to save rules as JSON files and generate executable Python validation functions. This view enables interactive exploration and modification of validation rules. It helps users understand both rules and validation results, supporting the iterative refinement process.

Validation rule visualization. Based on our rule classification (section III-B) and requirements analysis (section III-C), we developed a visualization approach inspired by MatrixWave [16]. The design supports our rule classification and represents both validation rules and their results. For cell-level rules, we use basic charts (area charts and histograms). For dependency rules between data cells, we employ a matrix-based visualization with basic charts along the edges (fig. 3A1) and relationship indicators in the matrix body (fig. 3A2).

The basic charts in our system vary by data type: (1) area charts for continuous data (numeric and datetime) with frequencies on the y-axis and values on the x-axis, and (2) histograms for discrete data (character) with categories in descending frequency order. For cell-level rules, these charts appear as standalone visualizations, with icons integrated for atomic rules (integrity, duplicate, and order), as illustrated in fig. 4A. When visualizing dependency rules, these charts are utilized as matrix edges (fig. 3A1) in our matrix visualization.

For dependency rules, we employ a matrix-based visualization. This approach represents the relationships between multiple data cells. The matrix edges consist of basic charts representing the data involved in the dependency. For column dependencies, different edges represent different columns; for row dependencies, edges represent the same column but in different row contexts (e.g., current row vs. next row values). The matrix body encodes the relationships between data elements on the edges. Its visual representation adapts to the data types: (1) discrete-discrete combinations create distinct regions (fig. 3A2), (2) continuous-continuous combinations appear as scatter plot-like data points (fig. 6B), and (3) discrete-continuous combinations display rectangular regions showing each discrete value's relationship with the continuous variable (fig. 4A1). For column-to-column dependencies (e.g., mapping), the matrix shows relationships between values from different columns. In fig. 5A, the highlighted region represents entries with "king" county and "bothell" city. For row-to-row dependencies (e.g., sequence), the matrix represents consecutive row relationships within a column. fig. 3A shows a `ball_status` sequence rule where the highlighted point connects "give pass" on the left edge to "get pass" on the right edge, indicating this pattern in consecutive rows. Complex validation scenarios may involve dependencies across multiple data elements. For example, validating `Postal Code` needs both `County` and `City`, as neither alone suffices due to duplicate names or cross-county cities. We connect multiple matrices through shared edges (fig. 4A2) to help users understand dependencies across multiple data elements.

The validation rule visualization employs a consistent color encoding: blue represents data that complies with validation rules, whereas orange indicates rule violations. In fig. 3A2, orange cells indicate rule-violating sequences between "give pass" and "get pass", while in fig. 6B, blue regions show valid numerical ranges. Additionally, we use yellow to represent intermediate states in multi-variable relationships, where a data point connects to both valid and invalid data. For instance, when visualizing relationships among `County`, `City`, and

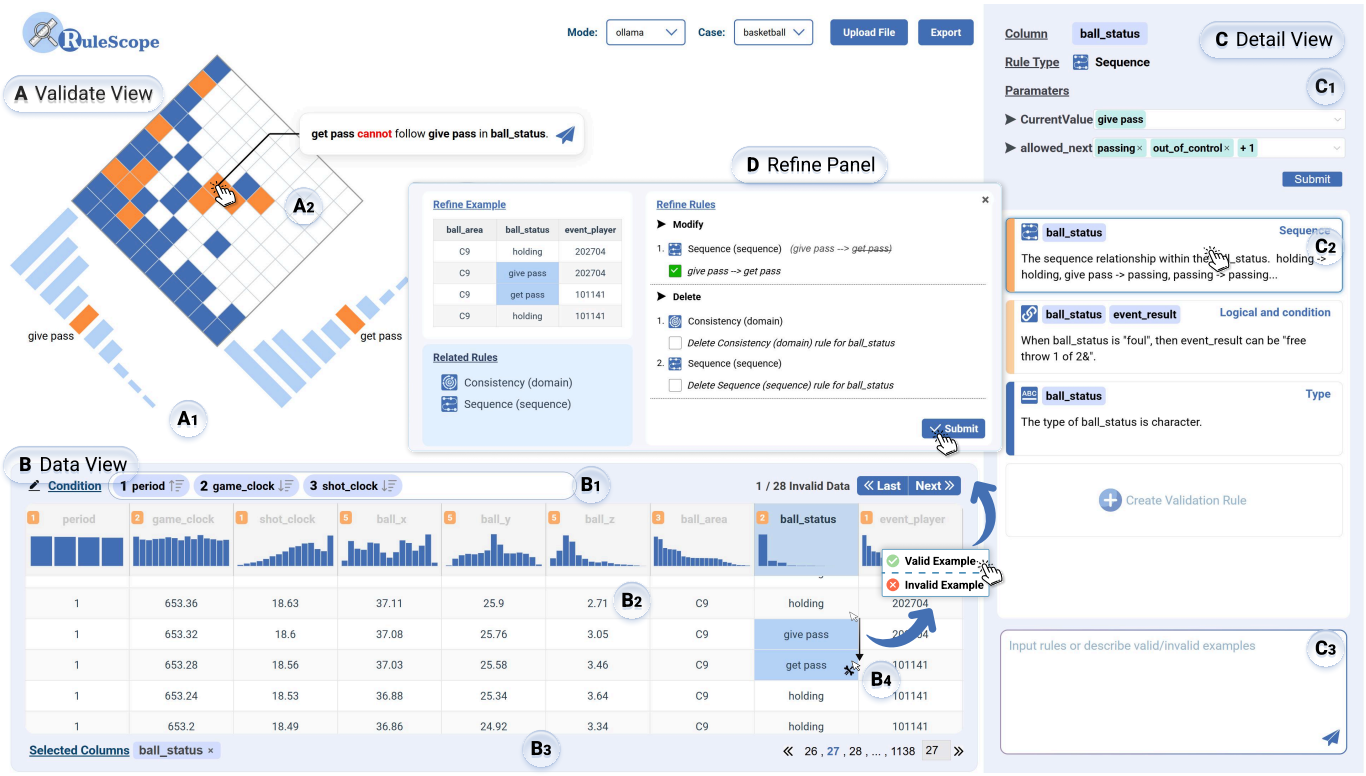

Fig. 3. The interface of RuleScope. (A) Validate View contains a matrix-based visualization of validation rules and their results; (B) Data View displays detailed data and corresponding validation results; (C) Detail View includes a setting panel for validation rules (C1), validation card panel that presents an overview of validation rules (C2), and natural language interaction interface (C3).

`Postal Code` columns (fig. 4A2), a connection point between "bothell" and "snohomish" appears yellow because it links to both valid postal codes (e.g., 98021) and invalid ones (e.g., 98258). For rules represented by icons (integrity, duplicate, and order), blue icons indicate validation rule presence and orange icons show violations (fig. 4A).

Interactions. The Validate View supports interactive exploration and refinement of validation rules. For exploration, users can click on any value along one matrix edge, which displays its label and highlights related values on the opposite edge. These highlights follow the system's color encoding (blue for valid, orange for invalid, yellow for intermediate states). Additionally, clicking within the matrix body reveals natural language explanations of the validation rules while simultaneously highlighting relevant data entries in the Data View. For example, as shown in fig. 3A2, clicking a violation region in the `ball_status` sequence matrix reveals the rule explanation and rule-violating data entries. The Validate View also enables rule refinement through several interaction methods. Users can edit rules by modifying the natural language explanations. Alternatively, they can right-click on specific matrix regions to adjust rule parameters, converting valid data to invalid or vice versa.

Design alternatives. During the design phase, we evaluated multiple approaches to visualize validation rules and their results. Through an iterative design process, we assessed parallel coordinates plots, Sankey diagram, chord chart, and matrix-based visualizations. Initial analysis showed that Sankey diagram (fig. 4B1) and chord chart (fig. 4B2) effectively visualized relationships between variables. For example, they could intuitively represent sequence rules by showing clear data flow patterns. However, testing with complete datasets and comprehensive rule sets revealed significant limitations. Overlapping lines and flow bands created excessive visual clutter, compromising rule visibility and increasing cognitive load from multiple variable representations. More critically, these methods could not effectively represent all validation rule types. Additionally, these methods inadequately support multi-variable dependencies (e.g., col C depends on col A and col B): Sankey diagrams and parallel coordinates struggle to encode continuous numerical constraints without binning, while chord charts cannot represent multi-variable relationships beyond pairwise connections. Considering visualization consistency and reducing learning overhead, we adopted the matrix-based approach with multi-matrix to represent multi-variable dependencies, augmented with in-situ explanation capabilities and integrated data navigation.

B. Data View

The Data View (fig. 3B) displays uploaded datasets and facilitates data exploration. The upper section (fig. 3B1) presents rule conditions (e.g., temporal ordering) on the left side, while the right side features navigation controls that help users track and locate invalid data entries. The central component is an interactive data table (fig. 3B2) with histogram-enhanced column headers showing data distributions. Each column header

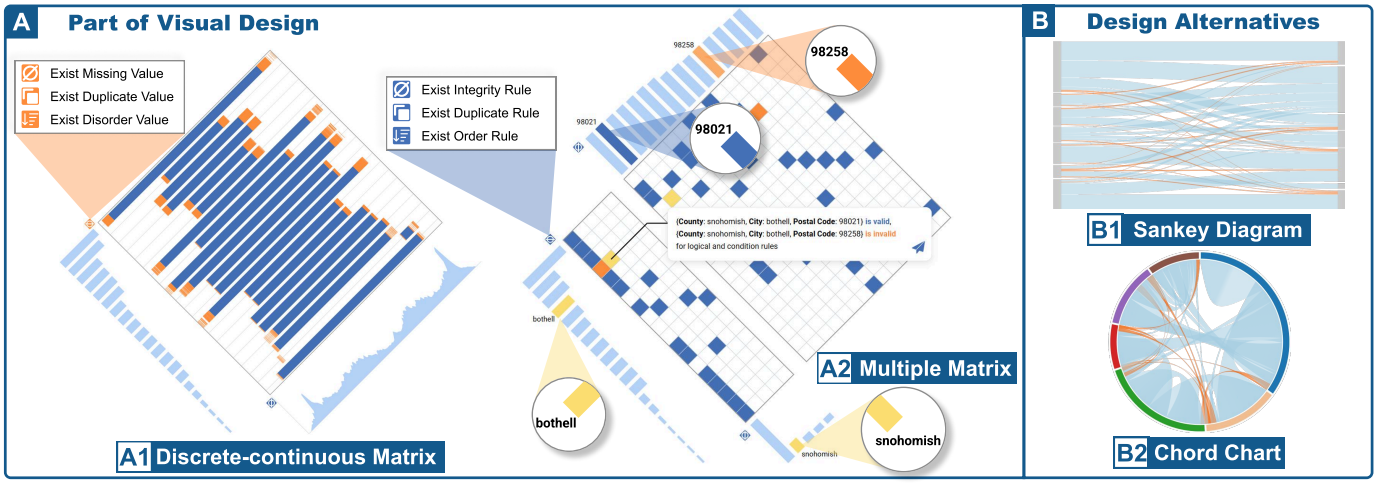

Fig. 4. Part of the visual design (A) and design alternatives (B) of RuleScope. (A) Matrix visualizations and icons representing atomic rules: a matrix composed of discrete-continuous edges (A1); multi-matrix visualization for validation rules involving multiple columns (A2). (B) Design alternatives: Sankey diagram (B1) and Chord chart (B2) as design alternatives for representing validation rules during our design process.

also indicates the number of rule violations associated with that column on its left side. The table highlights and provides navigation to relevant data through coordination with the Validate View and Detail View. The lower section (fig. 3B3) provides column selection indicators and pagination controls.

Interactions. Users can click column headers to filter rules relevant to specific data attributes. The Detail View then shows only rules associated with selected columns, while unrelated columns become grayed out and unselectable. Users can deselect columns by clicking headers again or by modifying the selection indicators. For data filtering, users can right-click histogram bars in column headers to filter data to the selected range. When a validation rule is selected in the Detail View, its conditions appear in the upper-left area with an edit option, while the table content is automatically sorted based on these conditions. The upper-right area shows the number of invalid data detected by the rule and provides navigation controls to locate these invalid data.

Users can select data regions in the data table and mark them as valid or invalid examples to refine validation rules. The Validation Rule Refiner (section IV-C) analyzes both selection patterns (cell, inter-row, inter-column, or multi-dimensional) and data characteristics to generate refinement options. RuleScope then presents applicable validation rules through a Refine Panel (fig. 3D). For example, a user marks adjacent "give pass" and "get pass" cells as valid (fig. 3B4), the Refine Panel displays the corresponding rules for refinement.

Refine Panel. The Refine Panel (fig. 3D) facilitates rule refinement by presenting analysis results based on user-selected data. The panel consists of three sections: the upper-left displays selected data patterns, the lower-left presents related validation rules, and the right section categorizes available refinements. The right section categorizes refinements as add, modify, or delete operations, helping users understand potential rule changes. To implement refinements, users can select desired changes and apply them through the submit button.

C. Detail View

The Detail View (fig. 3C) enables users to examine and modify validation rules through three integrated panels. The setting panel (fig. 3C1) displays parameters of selected validation rules and enables direct modification. The validation card panel (fig. 3C2) provides an overview of all rules and their validation status. The natural language panel (fig. 3C3) supports rule refinement through natural language input.

Validation card. Each validation card is divided into two sections. The upper section displays a rule type icon and column names on the left, with the rule type on the right. The lower section presents rule explanation and valid data samples. As shown in fig. 3C2, this validation card represents a sequence rule for the `ball_status` column. A color indicator along the card's left edge signals rule compliance. Blue shows all data conforms to the rule, while orange indicates violations, with darker shades reflecting higher percentages of invalid data. Upon detection of violations, validation cards are positioned at the top of the panel and sorted in descending order of violation percentage.

Interactions. Users can inspect rule details by clicking on validation cards, which triggers three coordinated actions: the Validate View displays the rule's visualization, the Data View highlights invalid data in orange, and the setting panel shows the current rule parameters. For multi-matrix layouts, users can drag-and-drop the column list in the setting panel to reorder elements and examine different pairwise relationships. For rule refinement, users can modify parameters through the setting panel, or leverage the natural language interface, where the LLM translates descriptions into validation rules.

VI. Evaluation

In this section, we evaluate the LLM-based workflow through model evaluation (section VI-A) and assess RuleScope's usability and effectiveness through two case studies (section VI-B) and a user study (section VI-D).

A. Model Evaluation

This section evaluates the LLM-based workflow (section IV) to examine whether the generated rules can effectively assist users in establishing initial validation rule sets. We adopted a non-comparative evaluation for two reasons. First, we focus on the workflow's utility rather than comparing performance across methods. Second, existing rule generation algorithms vary significantly in supported rule types, granularity, representation formats, and data requirements, making comparison difficult (see Sec. 3.1 of supplementary materials for details).

The evaluation was conducted on 10 datasets collected from four domains, with descriptions and sources provided in the supplemental material. These datasets varied in size (Rows: 1,320–1,048,575; Cols: 6–23). To establish a ground truth for evaluation, we invited domain experts to formulate validation rules for the datasets. This process yielded a total of 413 validation rules, comprising 322 cell-level rules and 91 dependency rules (see Sec. 3.2 of supplementary materials for construction details).

We assessed our approach using three metrics averaged across 10 rounds: rule type precision and recall, parameter coverage, and rule generation time (see Suppl. Sec. 2.3 for setup). First, rule type precision and recall measured the ability to identify correct rules against ground truth (e.g., identifying sequence validation for `ball_status`). Our workflow achieved an overall precision of 85.5% (SD=0.83%) and recall of 91.8% (SD=0.95%). Specifically, cell-level rules achieved a precision of 94.7% (SD=0.84%) and a recall of 93.0% (SD=1.24%). Dependency rules achieved a precision of 62.7% (SD=2.32%) and a recall of 87.7% (SD=1.45%). Second, parameter coverage quantified overlap with ground truth parameters. For instance, in a sequence rule for the `ball_status` column, a parameter might specify that "give pass" must be followed by "passing" or "out_of_control". The overall parameters coverage rate was 90.6% (SD = 1.06%), with cell-level rules achieving 93.2% (SD = 0.86%) and dependency rules reaching 80.8% (SD = 2.01%). Finally, the average rule generation time was 8.199s (SD=0.928s). Specifically, the generation time was 3.393s (SD=0.145s) for cell-level rules and 20.117s (SD=2.942s) for dependency rules. We further analyzed these metrics to understand the workflow's behavior (see Sec. 3.4 of supplementary materials for details).

In summary, our LLM-based workflow effectively generates initial validation rules by leveraging data semantics and domain knowledge. However, results show that LLMs still struggle with certain scenarios. These limitations highlight that validation rules still require expert review and refinement. Visualization is needed to represent rules and results for expert comprehension, while flexible interactions are required to support efficient rule refinement for accurate data validation.

B. Case Study

To evaluate the effectiveness and usability of RuleScope in data validation tasks, we conducted two case studies with four experts (E2, E4, E5, E6). While E2 and E4 were experts we collaborated with in our requirement analysis (section III-C), E5 and E6 were newly recruited for these case studies. The four experts consisted of two data analysts (E2, E5) and two PhD candidates (E4, E6), each with over three years of professional experience in their domains. The case studies demonstrated RuleScope's capability in validation rule authoring for data validation tasks with data from different domains.

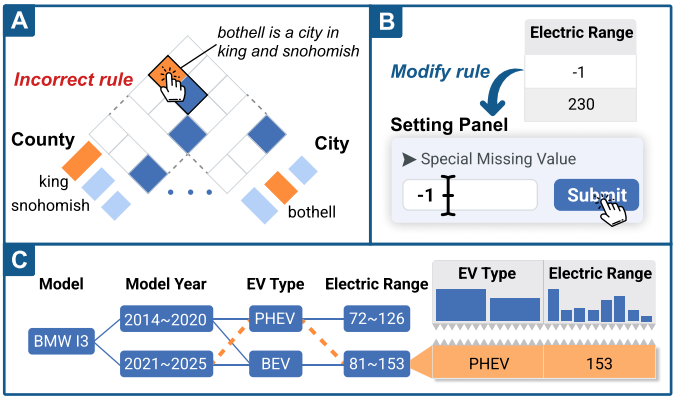

Fig. 5. Case 1 validation process: (A) Incorrect rules misidentify correct data as errors; (B) User refines rules to properly detect special missing values; (C) Rules detected data records with incorrectly marked EV Type.

1) Data and procedure: We prepared two datasets from different domains. Case 1 employed a U.S. electric vehicle (EV) registration dataset with attributes such as manufacturer, model, production year, and region. Case 2 utilized a basketball event dataset containing game information such as ball position, ball status, events, and associated players, derived from video frames captured at 0.04-second intervals.

The case studies were structured in three phases, totaling 50 minutes. The first phase (10 minutes) consisted of a tutorial session where experts learned how to read validation rule visualizations and refine rules through different interactions. In the second phase (20 minutes), based on their domain expertise, E2 and E5 validated the EV registration dataset (Case 1), while E4 and E6 worked on the basketball event dataset (Case 2). As all experts produced similar validation rules, due to space limitations, we document the processes of E4, E5 as representatives of their respective cases. The final phase (20 minutes) consisted of semi-structured interviews to collect experts' feedback.

2) Validating the Electric Vehicle Dataset: After RuleScope generated and applied the validation rules to the dataset, E5 noticed alerts in the Data View indicating issues in the `City` column. Since the subsequent analysis would involve examining EV registrations by location, E5 decided to investigate these violations by selecting the `City` column to examine the underlying validation rules. The Detail View then revealed a violation in the `City-County` mapping validation rule. By clicking the validation card, the Validate View displayed the matrix visualization of the rule (fig. 5A), which showed that "bothell" was mapped to both "king" and "snohomish" counties in the dataset, while the rule only allowed mapping to "snohomish". The data was actually correct, as "bothell" is a cross-county city, indicating that the validation rule required refinement. E5 selected the violating data in the Data View and marked them as valid examples, as these entries accurately

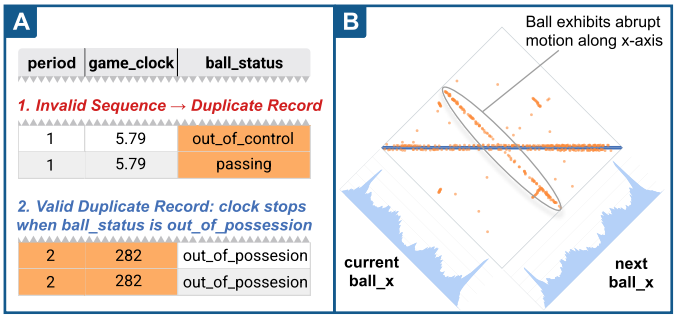

Fig. 6. Part of case 2 validation: (A) Identified duplicate records through sequence rule; (B) Detected irregular position changes in ball_x.

reflected the cross-county nature of "bothell". Based on RuleScope's recommendations, she updated the mapping rule by adding "bothell" to the "king" county mapping configuration.

Subsequently, E5 noticed alerts indicating violations in the `Electric Range` column, a critical metric for analyzing vehicle models and their range performance. She clicked on the validation card for the range validation rule (constraining values between 1 and 400 miles), which flagged several violating entries. Upon inspection, these values were predominantly -1, which were coded as missing data during vehicle registration. However, the missing value validation rule for the `Electric Range` had failed to detect them. To address this issue, she configured the missing value validation rule through the Setting Panel in the Detail View to designate -1 as a special missing value (fig. 5B), enabling proper validation.

Finally, E5 examined the rules for the relationship between electric range values and vehicle types (BEV and PHEV), which would be essential for her subsequent analysis. In the Detail View, she identified a logic and condition rule validating the dependency between `EV Type` and `Electric Range`, which flagged several violating entries. This rule detected multiple violations in the current dataset. Through the matrix visualization, E5 discovered several PHEV vehicles exceeding the expected range limit of 1-130 miles for PHEVs. E5 clicked on the orange-highlighted area in the matrix, which displayed the corresponding entries in the Data View. Examining these highlighted violations revealed a specific pattern: several 2020 BMW I3 models were recorded with 153-mile ranges (fig. 5C). This classification was confirmed to be incorrect, as BMW had discontinued the range-extender (PHEV) variant of the I3 by 2020, indicating that these vehicles should have been correctly classified as BEV models. The validation rule thus helped E5 identify these classification errors in the data.

3) Validating the Basketball Event Dataset: After RuleScope performed the initial validation on the dataset, E4 reviewed the validation cards in Detail View and identified multiple rule violations. As the subsequent analysis tasks focused on basketball event data, E4 first filtered the validation rules by clicking the `ball_status` column in Data View, and then examined these rules in detail.

E4 identified a sequence validation rule detecting non-compliant data in Detail View. Upon clicking the validation card, a matrix visualization displaying sequence rules appeared in Validate View. E4 clicked matrix regions to view rule descriptions, then refined rules by modifying descriptions or right-clicking on matrix areas. During this refinement process, he noticed that a "give pass" was directly followed by a "get pass", which violated the expected pattern where a "passing" or "out_of_control" state should occur between them. E4 clicked the invalid region in matrix (fig. 3A2) and the related data were highlighted in Data View. Analysis of these rows revealed a consistent temporal progression in `game_clock`, with continuous changes in ball position (`ball_x`, `ball_y`, `ball_z`). These patterns indicated a special case of hand-to-hand passing. Recognizing this as a valid exception, E4 selected the data and marked it as "valid example" (fig. 3B4). RuleScope generated refined rules in Refine Panel (fig. 3D), then E4 selected the appropriate refinement and submitted it.

During sequence error inspection, E4 identified another invalid sequence where "passing" followed "out_of_control". Investigating these records in the Data View, E4 discovered alternating "out_of_control" and "passing" (fig. 6A1). Analysis of `game_clock` and `period` showed duplicate records with different ball states at identical timestamps. As each timestamp should typically correspond to a single record, E4 clicked `game_clock` and found a validation rule requiring that the combination of `game_clock` and `period` should be unique across records. By clicking the validation card and reviewing invalid data, he identified two distinct scenarios: the previously detected duplicate records and a special case where the `game_clock` stops during "out_of_possession" (fig. 6A2). Since the latter represented valid situations, E4 deleted this inapplicable rule.

Since the subsequent analysis would require basketball movement trajectory data. E4 examined the basketball's velocity constraints by checking the differences between consecutive positions along the x, y, and z axes. He observed that rule violations were concentrated above and below the valid range band, while showing sparse occurrences at the matrix edges, indicating potentially over-restrictive validation rules. To verify this hypothesis, E4 clicked on invalid data points and examined them in the Data View alongside related columns. After confirming the rules were indeed too strict, E4 refined the difference-type validation rule for `ball_x` by selecting valid and invalid examples in the Data View, achieving a better balance between rule strictness and accuracy. During this refinement process, he discovered a distinct pattern in the matrix visualization of the `ball_x` column: a vertical concentration of rule violations near the court's centerline, demonstrating abrupt shifts along x-axes (fig. 6B). Investigation in the Data View showed these violations originated from dataset preprocessing. The preprocessing had removed all backcourt data, causing the ball to suddenly appear at the centerline. Understanding this dataset characteristic, E4 decided to segment the dataset by shot clock periods before conducting event sequence analysis.

C. Expert Interview

After the experts completed the rule authoring tasks, we conducted semi-structured interviews to gather their feedback on the system's effectiveness, usability, and improvements, as well as comparisons with their existing validation approaches.

Comparison with existing approaches. All experts agreed that RuleScope was more efficient than their existing tools, and noted that it reduced context-switching costs, as their prior workflows often required moving between multiple tools. Experts using interactive tools such as Excel (E2, E5) highlighted the contrast in workflow. E5 noted, *"With Excel, I have to create charts to visualize the data and then write formulas separately for validation. With RuleScope, I can inspect and refine validation rules with much less effort."* E2 also noted that *"RuleScope supports rule import and export, making it easy to reuse previous rules when validating similar datasets."* Experts relying on scripting-based validation (E2, E4, E6) found the process time-consuming, as rules had to be coded individually and results reviewed from lengthy outputs. E4 noted that RuleScope's visual presentation of rules alongside validation results reduced much of the effort in data validation. E4 also stated, *"when refining rules, I can directly select data instances from the table, which is more convenient than modifying parameters in code."* However, several experts noted a learning curve due to the number of available features. Nonetheless, all experts agreed that the efficiency gains in their actual workflows justified the initial learning effort.

Effectiveness and system design. All experts agreed that RuleScope improved their efficiency in data validation tasks. They found the majority of recommended rules aligned with their validation requirements, reducing the difficulty of creating rules from scratch. E2 and E5 mentioned that reviewing rule correctness stimulated their deeper thinking about validation rules, inspiring rule improvements for the dataset. E5 and E6 praised the effectiveness of the matrix design and natural language interaction. E5 stated, *"The matrix provides me with an initial overview of validation rules, while the natural language explanations enable me to understand them in detail."* Experts appreciated RuleScope's multiple interaction modes for rule refinement. When they had specific modification requirements, they could interact with visualization or modify natural language descriptions, *"... after reading the rule descriptions, directly modifying rules through editing these descriptions is seamless, which aligns perfectly with my workflow"* as E6 mentioned. For uncertain cases requiring data pattern examination, they could refine rules by selecting data instances. Experts highlighted RuleScope's functionality covered their entire data validation workflow, eliminating the need to switch between different tools.

Usability and improvements. Experts agreed on RuleScope's high usability, supporting their validation workflow, provided *"intuitive"* visualizations of rules and validation results, and offered *"convenient"* rule refinement. However, some experts suggested improvements to the rule generation process. E4 noted that *"some recommended rules are non-existent, and I have to spend extra time reviewing them."* He suggested adding a feature to adjust the diversity of rule recommendations, similar to the temperature parameter in LLMs, enabling adaptations to different phases of data validation. Regarding rule exploration, E2 and E6 suggested adding a zoom-in feature for matrices with continuous values, where areas with small coverage are difficult to analyze. This would allow users to select and magnify specific regions for better data exploration. E4 suggested implementing a search feature where users could express their validation requirements in natural language and directly locate the corresponding rules. For the refinement process, E5 pointed out, *"The current modification granularity is relatively fine-grained, allowing changes to only one rule or specific rule parameters at a time."* She suggested supporting the upload of data specification documents to generate modifications for multiple rules simultaneously. She also recommended providing detailed comparisons of rules before and after modifications to help users understand the differences.

D. User Study

We conducted a task-based user study to evaluate RuleScope's usability, measuring task completion time and success rates. We adopted a non-comparative evaluation for two reasons. First, to the best of our knowledge, no existing data validation tools adequately support validation rule authoring, especially for dependency rules. These tools lack capabilities for generating, visualizing, and refining rules (see Sec. 4.1 of the supplementary material for details). Second, our participants came from diverse professional domains with varying technical backgrounds, ranging from interactive spreadsheet tools to script-based approaches. This heterogeneity made it difficult to identify a suitable baseline tool for comparison.

1) User Study Settings: **(1) Participants and Data.** We recruited 16 participants (P1-P16) with a gender distribution of 6 females and 10 males from diverse academic backgrounds: Computer Science (9), Sports Training (2), Education (1), Digital Media (1), Management (1), Mathematics (1), and Business (1). All participants had prior experience in data validation, and none have been involved in the development of RuleScope. In this study, we used the same datasets (EV dataset and basketball dataset) from our case studies (see section VI-B). **(2) Procedure and Tasks.** The user study consisted of three phases and lasted approximately 60 minutes. Phase 1 was a 20-minute training session where participants learned about RuleScope's validation rule types, visualization, interaction, and domain knowledge for the study datasets. Phase 2 required participants to complete 12 tasks (please refer to supplementary material for details), with 6 tasks for each dataset. Following the data validation workflow, these tasks were divided into three stages (two tasks per stage) to evaluate different RuleScope functionalities: (1) understanding validation rules and results (R2); (2) analyzing validation results through detailed data examination (R3); and (3) refining validation rules (R4). Task completion times and performance were recorded. In Phase 3, participants completed a System Usability Scale (SUS) [87] questionnaire using a 5-point Likert scale and follow-up interviews for qualitative feedback. This user study has received approval from State Key Lab of CAD&CG, Zhejiang University.

2) Quantitative Results: We evaluated user performance with RuleScope by measuring task completion duration and success rates. Task success was determined based on appropriate operations in the system and correct answers.

The study reached a 0.99 task success rate, with only two recorded failures in T11. This task required examin-

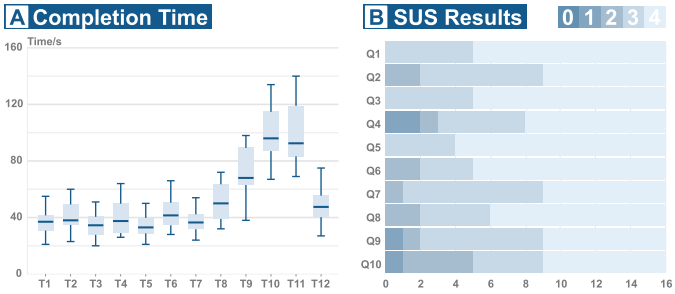

Fig. 7. Quantitative results of the user study: Distribution of completion times across tasks; (B) Distribution of SUS usability ratings.

ing logic and condition rules validating the relationship between `ball_status` and `event_result` in the basketball dataset, evaluating these rules against domain knowledge, and modifying them. P2 and P3 failed for different reasons. P2 correctly identified and fixed invalid rules but introduced new errors by modifying initially correct rules. P3 failed to identify any rule errors due to misunderstanding the provided domain knowledge. For successfully completed tasks, completion times ranged from 20 to 141 seconds, averaging 53 seconds per task and 10.4 minutes per participant (see fig. 7A). Notably, T9-T11 exhibited longer completion times with greater variance, as these tasks required through analysis combining validation results and domain knowledge. Participants gave a SUS score of 85.9, exceeding the top 10% threshold of 80.3 [87]. Further analysis of SUS factors [88] revealed a usability score (Q1-Q3, Q5-Q9) of 87.9 and a learnability score (Q4 and Q10) of 78.1. The details of the participants' scores are shown in fig. 7B. User feedback helped explain these scores. The high usability score was attributed to the intuitive matrix visualization, while the lower learnability score reflected challenges with multiple interaction for refining validation rules. To address these challenges, future iterations will focus on improving interaction guidance. Overall, participants found RuleScope effective and expressed willingness to use it in their work.

3) Qualitative Results: This section presents a detailed analysis of participant feedback, revealing key insights about RuleScope's effectiveness in supporting data validation tasks and opportunities for improvement.

RuleScope effectively supported understanding of validation rules and results (R2). Most (13/16) participants emphasized how the matrix visualization facilitated their understanding of rules (10/16) and results (8/16). For example, participants mentioned that the matrix visualization provided *"clear natural language explanation of rules"* (P1, P3, P5, P10), helped them *"discover patterns in rules"* (P7, P11), and understand *"the proportion of invalid data"* (P1, P4, P5, P14) in the current dataset. The Detail View's validation cards provided effective rule overview and context-specific information. P7 mentioned that *"The validation card is intuitive, I can understand the rule after reading it."* While participants generally found the matrix visualization approachable, several noted a learning curve in interpreting its visual elements. P9, P12 and P15 reported needing initial effort to comprehend different rule types' visual representations. As P9 explained: *"Different validation rules show distinct matrix visualizations, and it took some time at first to understand these variations."* They suggested incorporating more in-situ guidance to help users learn the visual encodings.

RuleScope effectively supports the analysis of validation rules and results (R3). Participants (11/16) confirmed that the detailed data and statistical information in the Data View enabled them to effectively analyze both validation rules and results. During the discussion of the Data View, P10 noted: *"When analyzing rule appropriateness, I need to examine the full context of the data entry rather than isolated values, especially for corner cases..."* P13 suggested improvements regarding the Data View's current limitation of examining invalid data for only one validation rule at a time, which reduced their analysis efficiency. He recommended implementing functionality to simultaneously display validation results for multiple user-selected rules, enabling efficient comparison and identification of both validation rule and quality issues.

RuleScope's flexible interactions facilitate effective rule refinement (R4). Participants demonstrated distinct preferences for rule refinement methods, reflecting different mental models and workflows. Some participants (P1, P5, P12) favored natural language refinement, typically approaching rule modifications from a semantic perspective. Another group of participants (P4, P7, P10) preferred the by-example approach, relying on data analysis to guide their rule modifications. Other participants (P11, P14, P16) opted for direct parameter modification in the Setting Panel, finding it analogous to familiar tools, which enhanced their confidence during the modification process. Participants also highlighted areas for improvement. Regarding natural language, P8 expressed concerns that *"The system's interpretation of user intent might be incorrect, increasing refinement time."* She suggested that *"A confirmation step comparing original and modified rules could address this issue."* For the by-example, participants familiar with spreadsheet tools expressed concerns about the interaction paradigm. As P16 noted, *"The data selection operation made me feel like I needed to modify the data rather than refine the rules."* The diversity of refinement methods, while providing flexibility, introduced cognitive challenges. P6 and P9 noted that multiple methods increased learning burden and caused decision paralysis, suggesting context-aware guidance based on interaction views or rule types.

VII. Discussion

In this section, we discuss the implications, scalability, generalizability, limitations and future work of RuleScope.

Implications. This study presents a semantic-aware approach to data validation rule authoring through rule generation, verification, and interactive refinement. RuleScope implements validation rule generation powered by LLMs based on data semantics and domain knowledge, integrating data preprocessing at multiple granularity levels with targeted prompt engineering to guide LLMs in applying appropriate reasoning for different rule types. The matrix-based visualizations reveal patterns in both validation rules and results,

helping practitioners understand and refine rules. Our evaluation examines human-LLM collaboration in data validation tasks, analyzing how this collaborative approach inspires users and offering insights for enhancing human-AI cooperation in rule authoring. This integrated approach of LLMs, interactive visualizations, and collaborative workflows provides insights for developing more effective data validation systems.

Scalability. Our model evaluation demonstrates RuleScope's capability to support validation rule generation for tables with millions of rows and its adaptability to datasets from diverse domains. For visualization, our matrix-based approach effectively represents dependency rules compared to alternatives like Sankey diagrams, supporting users in exploring validation rules from overview to detail across multiple data entities. However, with large datasets, especially those with character columns containing many distinct values, matrix visualization density increases, making interpretation challenging. Although zooming functionality helps mitigate this issue, zooming into the matrix often loses contextual information at the edges, requiring repeated zoom adjustments. For future work, we propose local magnification through brush-selection to balance focus and context.

Generalizability. Validation rules authored in RuleScope can be effectively transferred to other tools and systems, enhancing cross-platform compatibility. Users can leverage the export functionality in the interface to obtain validation rules in JSON format or data validation functions in Python. For instance, in data cleaning workflows, developers can utilize RuleScope-exported Python functions to identify data entries that violate validation rules, facilitating targeted cleaning and transformation operations to meet downstream requirements. However, the current export options are limited to JSON and Python formats. Beyond export compatibility, RuleScope currently operates on tabular data. Semi-structured formats such as JSON can be partially accommodated by converting them into tabular form prior to upload. However, this conversion does not preserve hierarchical relationships in deeply nested structures, which the current system does not support. Future work could support user-defined templates for exporting rules and validation functions to better align with integration needs, and extend the validation workflow to natively accommodate semi-structured data across diverse formats.

Model Considerations. RuleScope is implemented with Qwen2.5 [85] for local deployment and supports different LLMs through API interfaces. However, rule quality may vary across models. Future work should leverage user refinement feedback to improve cross-model consistency. Regarding ethical considerations, we minimize the temperature parameter, carefully design prompts, and rely on robust ethical safeguards built into modern LLMs [85], [89]–[91]. Future work should introduce explicit ethical verification mechanisms to prevent potentially problematic validation logic.

Limitations and future work. We identified four main limitations in RuleScope. First, although the LLM-based workflow facilitates the generation of initial validation rules, these generated rules are not perfect and necessitate human intervention for refinement. In future work, we plan to enhance the overall rule generation process through several directions. To improve generation quality, we will optimize data feature extraction techniques. We also plan to implement mechanisms that enable users to provide dataset descriptions and validation requirements prior to rule generation. To help users identify potentially problematic rules, we plan to introduce cross-evaluation across multiple LLMs to estimate a confidence score for each generated rule. This mechanism will guide users toward low-confidence rules that warrant closer inspection, reducing review effort and enhancing overall trust in RuleScope. Second, the current rule verification and refinement process requires manual inspection of individual rules, which becomes labor-intensive when dealing with a large number of validation rules. To improve efficiency, we plan to introduce automated mechanisms that retrieve and suggest potential modifications for rules with similar patterns based on user refinement operations. Third, the system focuses on detection rather than integrated data cleaning, as the appropriate cleaning method depends on external factors like the underlying error causes and analytical task requirements. We plan to enable users to express cleaning intentions concurrently with rule authoring, utilizing these inputs to drive semantic-based algorithms that recommend appropriate correction methods to resolve invalid data. Fourth, our matrix visualization is limited in revealing all pairwise relationships when dealing with multiple columns. In future work, we plan to support automatic sorting to facilitate multi-column exploration.

VIII. Conclusion

This study presents RuleScope, an interactive system for authoring validation rules through semantic-aware rule generation, visualization, and refinement. RuleScope leverages an LLM-based workflow to generate interpretable validation rules by analyzing data semantics and incorporating domain knowledge. The system incorporates a novel matrix-based visualization for comprehension of validation rules and results, and supports interactive rule refinement. The evaluation demonstrates that RuleScope is effective in facilitating validation rule authoring across different domains, thereby enabling efficient data validation.

IX. Acknowledgments

The work was supported by NSFC (62532011, 62402421), Zhejiang Provincial Natural Science Foundation of China (LD25F020003), Key "Pioneer" R&D Projects of Zhejiang Province (2026C01027), and Ningbo Yongjiang Talent Programme (2024A-399-G). We gratefully thank the anonymous reviewers for their valuable comments.

References

[1] X. Chu, I. F. Ilyas, S. Krishnan, and J. Wang, "Data Cleaning: Overview and Emerging Challenges," in *Proceedings of the International Conference on Management of Data*, 2016, pp. 2201–2206.

[2] L. Li, T. Peng, and J. Kennedy, "A rule based taxonomy of dirty data." *GSTF Journal on Computing*, vol. 1, no. 2, pp. 140–148, 2011.

[3] N. Laranjeiro, S. N. Soydemir, and J. Bernardino, "A Survey on Data Quality: Classifying Poor Data," in *Proceedings of IEEE Pacific Rim International Symposium on Dependable Computing*, 2015, pp. 179–188.

[4] I. F. Ilyas and X. Chu, "Trends in Cleaning Relational Data: Consistency and Deduplication," *Foundations and Trends® in Databases*, vol. 5, no. 4, pp. 281–393, 2015.

[5] M. R. Wigan and R. Clarke, "Big data's big unintended consequences," *Computer*, vol. 46, no. 6, pp. 46–53, 2013.

[6] L. P. English, *Information Quality Applied: Best Practices for Improving Business Information, Processes and Systems*. Wiley Publishing, 2009.

[7] Susan Moore, "How To Create A Business Case For Data Quality Improvement," https://www.gartner.com/smarterwithgartner/how-to-create-a-business-case-for-data-quality-improvement, 2023.

[8] D. Tu, Y. He, W. Cui, S. Ge, H. Zhang, S. Han, D. Zhang, and S. Chaudhuri, "Auto-Validate by-History: Auto-Program Data Quality Constraints to Validate Recurring Data Pipelines," in *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023, pp. 4991–5003.

[9] J. Song and Y. He, "Auto-Validate: Unsupervised Data Validation Using Data-Domain Patterns Inferred from Data Lakes," in *Proceedings of the International Conference on Management of Data*, 2021, pp. 1678–1691.

[10] S. Redyuk, Z. Kaoudi, V. Markl, and S. Schelter, "Automating Data Quality Validation for Dynamic Data Ingestion," in *Proceedings of the International Conference on Extending Database Technology*, 2021, pp. 61–72.

[11] A. Fariha, A. Tiwari, A. Meliou, A. Radhakrishna, and S. Gulwani, "CoCo: Interactive Exploration of Conformance Constraints for Data Understanding and Data Cleaning," in *Proceedings of the International Conference on Management of Data*, 2021, pp. 2706–2710.

[12] M. Mahdavi, Z. Abedjan, R. Castro Fernandez, S. Madden, M. Ouzzani, M. Stonebraker, and N. Tang, "Raha: A Configuration-Free Error Detection System," in *Proceedings of the International Conference on Management of Data*, 2019, pp. 865–882.

[13] P. Wang and Y. He, "Uni-Detect: A Unified Approach to Automated Error Detection in Tables," in *Proceedings of the International Conference on Management of Data*, 2019, pp. 811–828.

[14] S. G. Powell, K. R. Baker, and B. Lawson, "A critical review of the literature on spreadsheet errors," *Decision Support Systems*, vol. 46, no. 1, pp. 128–138, 2008.

[15] R. A. Ruddle, J. Cheshire, and S. J. Fernstad, "Tasks and Visualizations Used for Data Profiling: A Survey and Interview Study," *IEEE Transactions on Visualization and Computer Graphics*, vol. 30, no. 7, pp. 3400–3412, 2024.

[16] J. Zhao, Z. Liu, M. Dontcheva, A. Hertzmann, and A. Wilson, "MatrixWave: Visual Comparison of Event Sequence Data," in *Proceedings of the Annual ACM Conference on Human Factors in Computing Systems*, 2015, pp. 259–268.

[17] N. Polyzotis, M. Zinkevich, S. Roy, E. Breck, and S. Whang, "Data Validation for Machine Learning," *Proceedings of Machine Learning and Systems*, vol. 1, pp. 334–347, 2019.

[18] TensorFlow, "TensorFlow Data Validation: Checking and analyzing your data | TFX," https://www.tensorflow.org/tfx/guide/tfdv.

[19] S. Schelter, F. Biessmann, D. Lange, T. Rukat, P. Schmidt, S. Seufert, P. Brunelle, and A. Taptunov, "Unit Testing Data with Deequ," in *Proceedings of the International Conference on Management of Data*, 2019, pp. 1993–1996.

[20] S. Schelter, D. Lange, P. Schmidt, M. Celikel, F. Biessmann, and A. Grafberger, "Automating large-scale data quality verification," *Proceedings of the VLDB Endowment*, vol. 11, no. 12, pp. 1781–1794, 2018.

[21] A. Swami, S. Vasudevan, and J. Huyn, "Data Sentinel: A Declarative Production-Scale Data Validation Platform," in *Proceedings of IEEE International Conference on Data Engineering*, 2020, pp. 1579–1590.

[22] Data Sentinel, "Sensitive Data Management - Data Sentinel," https://www.data-sentinel.com/.

[23] Microsoft, "Microsoft Excel Spreadsheet Software," https://www.microsoft.com/en-us/microsoft-365/excel.

[24] A. Wyatt, "Microsoft Excel error checking rules," https://excelribbon.tips.net/T006221_Changing_Error_Checking_Rules.html.

[25] Trifacta, "The Trifacta Data Engineering Cloud," https://www.trifacta.com/.

[26] Microsoft, "Power BI - Data Visualization | Microsoft Power Platform," https://www.microsoft.com/en-us/power-platform/products/power-bi.

[27] Talend, "Talend | A Complete, Scalable Data Management Solution," https://www.talend.com/.

[28] OpenRefine, "Openrefine (previously google refine)." https://openrefine.org/.

[29] C. C. Aggarwal and P. S. Yu, "Outlier detection for high dimensional data," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2001, pp. 37–46.

[30] S. Basu and M. Meckesheimer, "Automatic outlier detection for time series: An application to sensor data," *Knowledge and Information Systems*, vol. 11, no. 2, pp. 137–154, 2007.

[31] I. Ben-Gal, "Outlier Detection," in *Data Mining and Knowledge Discovery Handbook*. Springer US, 2005, pp. 131–146.

[32] Z. Ferdousi and A. Maeda, "Unsupervised Outlier Detection in Time Series Data," in *Proceedings of International Conference on Data Engineering Workshops*, 2006, pp. x121–x121.

[33] Z. Huang and Y. He, "Auto-Detect: Data-Driven Error Detection in Tables," in *Proceedings of the International Conference on Management of Data*, 2018, pp. 1377–1392.

[34] L. Berti-Équille, H. Harmouch, F. Naumann, N. Novelli, and S. Thirumuruganathan, "Discovery of genuine functional dependencies from relational data with missing values," *Proceedings of the VLDB Endowment*, vol. 11, no. 8, pp. 880–892, 2018.

[35] F. Chiang and R. J. Miller, "Discovering data quality rules," *Proceedings of the VLDB Endowment*, vol. 1, no. 1, pp. 1166–1177, 2008.

[36] X. Chu, I. F. Ilyas, and P. Papotti, "Discovering denial constraints," *Proceedings of the VLDB Endowment*, vol. 6, no. 13, pp. 1498–1509, 2013.

[37] T. Dasu, T. Johnson, S. Muthukrishnan, and V. Shkapenyuk, "Mining database structure; or, how to build a data quality browser," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2002, pp. 240–251.

[38] J. Kivinen and H. Mannila, "Approximate Inference of Functional Dependencies from Relations," *Theoretical Computer Science*, vol. 149, no. 1, pp. 129–149, 1995.

[39] A. Heidari, J. McGrath, I. F. Ilyas, and T. Rekatsinas, "HoloDetect: Few-Shot Learning for Error Detection," in *Proceedings of the International Conference on Management of Data*, 2019, pp. 829–846.

[40] A. Qahtan, N. Tang, M. Ouzzani, Y. Cao, and M. Stonebraker, "AUTOMAT: Automatic Knowledge Discovery and Error Detection through Pattern Functional Dependencies," in *Proceedings of the International Conference on Management of Data*, 2019, pp. 1977–1980.

[41] J. N. Yan, O. Schulte, M. Zhang, J. Wang, and R. Cheng, "SCODED: Statistical Constraint Oriented Data Error Detection," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2020, pp. 845–860.

[42] Z. Liu, Z. Zhou, and T. Rekatsinas, "Picket: Self-supervised data diagnostics for ML pipelines," *CoRR*, vol. abs/2006.04730, 2020.

[43] S. Kandel, R. Parikh, A. Paepcke, J. M. Hellerstein, and J. Heer, "Profiler: Integrated statistical analysis and visualization for data quality assessment," in *Proceedings of the International Working Conference on Advanced Visual Interfaces*, 2012, pp. 547–554.

[44] W. Willett, B. A. Aseniero, S. Carpendale, P. Dragicevic, Y. Jansen, L. Oehlberg, and P. Isenberg, "Perception! Immersion! Empowerment! Superpowers as Inspiration for Visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 1, pp. 22–32, 2022.

[45] W. Epperson, V. Gorantla, D. Moritz, and A. Perer, "Dead or Alive: Continuous Data Profiling for Interactive Data Science," *IEEE Transactions on Visualization and Computer Graphics*, vol. 30, no. 1, pp. 197–207, 2024.

[46] M. Dallachiesa, A. Ebaid, A. Eldawy, A. Elmagarmid, I. F. Ilyas, M. Ouzzani, and N. Tang, "NADEEF: A commodity data cleaning system," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2013, pp. 541–552.

[47] A. Fariha, A. Tiwari, A. Radhakrishna, S. Gulwani, and A. Meliou, "Conformance Constraint Discovery: Measuring Trust in Data-Driven Systems," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2021, pp. 499–512.

[48] S. Kandel, A. Paepcke, J. Hellerstein, and J. Heer, "Wrangler: Interactive visual specification of data transformation scripts," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2011, pp. 3363–3372.

[49] T. Gschwandtner, W. Aigner, S. Miksch, J. Gärtner, S. Kriglstein, M. Pohl, and N. Suchy, "TimeCleanser: A visual analytics approach for data cleansing of time-oriented data," in *Proceedings of the International Conference on Knowledge Technologies and Data-driven Business*, 2014, pp. 1–8.

[50] Z. Zhou, Y. Li, Y. Ni, W. Xu, G. Hu, Y. Lai, P. Chen, and W. Su, "VisCI: A visualization framework for anomaly detection and interactive optimization of composite index," *Visual Informatics*, vol. 8, no. 2, pp. 1–12, 2024.

[51] C. Chai, L. Cao, G. Li, J. Li, Y. Luo, and S. Madden, "Human-in-the-loop Outlier Detection," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2020, pp. 19–33.

[52] S. Krishnan, J. Wang, E. Wu, M. J. Franklin, and K. Goldberg, "ActiveClean: Interactive data cleaning for statistical modeling," *Proceedings of the VLDB Endowment*, vol. 9, no. 12, pp. 948–959, 2016.
[53] T. Tang, Y. Wu, J. Gao, K. Ruan, Y. Zhang, S. Ye, Y. Wu, and X. Chen, "ArtEyer: Enriching GPT-based agents with contextual data visualizations for fine art authentication," *Visual Informatics*, vol. 8, no. 4, pp. 48–59, 2024.
[54] M. Bilgic, L. Licamele, L. Getoor, and B. Shneiderman, "D-Dupe: An Interactive Tool for Entity Resolution in Social Networks," in *Proceedings of the IEEE Symposium On Visual Analytics Science And Technology*, 2006, pp. 43–50.
[55] Y. Luo, C. Chai, X. Qin, N. Tang, and G. Li, "Interactive Cleaning for Progressive Visualization through Composite Questions," in *Proceedings of the IEEE International Conference on Data Engineering*, 2020, pp. 733–744.
[56] ——, "VisClean: Interactive cleaning for progressive visualization," *Proceedings of the VLDB Endowment*, vol. 13, no. 12, pp. 2821–2824, 2020.
[57] S. Gratzl, A. Lex, N. Gehlenborg, H. Pfister, and M. Streit, "LineUp: Visual Analysis of Multi-Attribute Rankings," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 12, pp. 2277–2286, 2013.
[58] J. Wu, D. Liu, Z. Guo, Q. Xu, and Y. Wu, "TacticFlow: Visual Analytics of Ever-Changing Tactics in Racket Sports," *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 1, pp. 835–845, 2022.
[59] Z. Deng, H. Chen, Q.-L. Lu, Z. Su, T. Schreck, J. Bao, and Y. Cai, "Visual comparative analytics of multimodal transportation," *Visual Informatics*, vol. 9, no. 1, pp. 18–30, 2025.
[60] Y. Wu, J. Lan, X. Shu, C. Ji, K. Zhao, J. Wang, and H. Zhang, "iTTVis: Interactive Visualization of Table Tennis Data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 709–718, 2018.
[61] S. Liu, C. Chen, Y. Lu, F. Ouyang, and B. Wang, "An Interactive Method to Improve Crowdsourced Annotations," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, pp. 235–245, 2019.
[62] C. Chen, J. Yuan, Y. Lu, Y. Liu, H. Su, S. Yuan, and S. Liu, "OoD-Analyzer: Interactive Analysis of Out-of-Distribution Samples," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 7, pp. 3335–3349, 2021.
[63] W. Yang, X. Ye, X. Zhang, L. Xiao, J. Xia, Z. Wang, J. Zhu, H. Pfister, and S. Liu, "Diagnosing Ensemble Few-Shot Classifiers," *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 9, pp. 3292–3306, 2022.
[64] W. Yang, Y. Guo, J. Wu, Z. Wang, L.-Z. Guo, Y.-F. Li, and S. Liu, "Interactive Reweighting for Mitigating Label Quality Issues," *IEEE Transactions on Visualization and Computer Graphics*, vol. 30, no. 3, pp. 1837–1852, 2024.
[65] C. Vajiac, D. H. Chau, A. Olligschlaeger, R. Mackenzie, P. Nair, M.-C. Lee, Y. Li, N. Park, R. Rabbany, and C. Faloutsos, "TrafficVis: Visualizing Organized Activity and Spatio-Temporal Patterns for Detecting and Labeling Human Trafficking," *IEEE Transactions on Visualization and Computer Graphics*, vol. 29, no. 1, pp. 53–62, 2023.
[66] C. Chen, J. Chen, W. Yang, H. Wang, J. Knittel, X. Zhao, S. Koch, T. Ertl, and S. Liu, "Enhancing Single-Frame Supervision for Better Temporal Action Localization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 30, no. 6, pp. 2903–2915, 2024.
[67] OpenAI, "Gpt-5.2," https://chatgpt.com/.
[68] Anthropic, "Claude 4.5," https://www.anthropic.com/.
[69] Y. Tian, W. Cui, D. Deng, X. Yi, Y. Yang, H. Zhang, and Y. Wu, "ChartGPT: Leveraging LLMs to Generate Charts From Abstract Natural Language," *IEEE Transactions on Visualization and Computer Graphics*, vol. 31, no. 3, pp. 1731–1745, 2025.
[70] H. W. Wang, M. Gordon, L. Battle, and J. Heer, "DracoGPT: Extracting Visualization Design Preferences from Large Language Models," *IEEE Transactions on Visualization and Computer Graphics*, vol. 31, no. 1, pp. 710–720, 2025.
[71] A. McNutt, M. C. Stone, and J. Heer, "Mixing Linters with GUIs: A Color Palette Design Probe," *IEEE Transactions on Visualization and Computer Graphics*, vol. 31, no. 1, pp. 327–337, 2025.
[72] T. S. Kim, D. Choi, Y. Choi, and J. Kim, "Stylette: Styling the web with natural language," in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 2022, pp. 1–17.
[73] C. Wang, J. Thompson, and B. Lee, "Data Formulator: AI-Powered Concept-Driven Visualization Authoring," *IEEE Transactions on Visualization and Computer Graphics*, vol. 30, no. 1, pp. 1128–1138, Jan. 2024.
[74] P. Maddigan and T. Susnjak, "Chat2VIS: Generating Data Visualizations via Natural Language Using ChatGPT, Codex and GPT-3 Large Language Models," *IEEE Access*, vol. 11, pp. 45 181–45 193, 2023.
[75] V. Dibia, "LIDA: A tool for automatic generation of grammar-agnostic visualizations and infographics using large language models," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 2023, pp. 113–126.
[76] M.-H. Hong and A. Crisan, "Conversational AI threads for visualizing multidimensional datasets," *arXiv preprint arXiv:2311.05590*, 2023.
[77] P. Ma, R. Ding, S. Wang, S. Han, and D. Zhang, "InsightPilot: An llm-empowered automated data exploration system," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2023, pp. 346–352.
[78] Z. Abedjan, X. Chu, D. Deng, R. C. Fernandez, I. F. Ilyas, M. Ouzzani, P. Papotti, M. Stonebraker, and N. Tang, "Detecting data errors: Where are we and what needs to be done?" *Proceedings of the VLDB Endowment*, vol. 9, no. 12, pp. 993–1004, 2016.
[79] M. Yakout, L. Berti-Équille, and A. K. Elmagarmid, "Don't be SCAREd: Use SCalable Automatic REpairing with maximal likelihood and bounded changes," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2013, pp. 553–564.
[80] J. Gao, C. Xie, and C. Tao, "Big Data Validation and Quality Assurance – Issuses, Challenges, and Needs," in *Proceedings of the IEEE Symposium on Service-Oriented System Engineering*, 2016, pp. 433–441.
[81] N. Hollmann, S. Müller, and F. Hutter, "Large Language Models for Automated Data Science: Introducing CAAFE for Context-Aware Automated Feature Engineering," *Advances in Neural Information Processing Systems*, vol. 36, pp. 44 753–44 775, 2023.
[82] Z. Huang and E. Wu, "Cocoon: Semantic Table Profiling Using Large Language Models," in *Proceedings of the 2024 Workshop on Human-In-the-Loop Data Analytics*, 2024, pp. 1–7.
[83] A. Narayan, I. Chami, L. Orr, and C. Ré, "Can Foundation Models Wrangle Your Data?" *Proceedings of the VLDB Endowment*, vol. 16, no. 4, pp. 738–746, 2022.
[84] S. Arora, B. Yang, S. Eyuboglu, A. Narayan, A. Hojel, I. Trummer, and C. Ré, "Language models enable simple systems for generating structured views of heterogeneous data lakes," *Proceedings of the VLDB Endowment*, vol. 17, no. 2, pp. 92–105, 2023.
[85] Qwen, A. Yang, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Li, D. Liu, F. Huang, H. Wei, H. Lin, J. Yang, J. Tu, J. Zhang, J. Yang, J. Yang, J. Zhou, J. Lin, K. Dang, K. Lu, K. Bao, K. Yang, L. Yu, M. Li, M. Xue, P. Zhang, Q. Zhu, R. Men, R. Lin, T. Li, T. Tang, T. Xia, X. Ren, X. Ren, Y. Fan, Y. Su, Y. Zhang, Y. Wan, Y. Liu, Z. Cui, Z. Zhang, and Z. Qiu, "Qwen2.5 Technical Report," 2025.
[86] Ollama, "Ollama," https://ollama.com.
[87] j. Brooke, "SUS: A 'Quick and Dirty' Usability Scale," in *Usability Evaluation In Industry*. CRC Press, 1996.
[88] J. R. Lewis and J. Sauro, "The Factor Structure of the System Usability Scale," in *Proceedings of the Human Centered Design*, 2009, pp. 94–103.
[89] D. Shi, T. Shen, Y. Huang, Z. Li, Y. Leng, R. Jin, C. Liu, X. Wu, Z. Guo, L. Yu, L. Shi, B. Jiang, and D. Xiong, "Large Language Model Safety: A Holistic Survey," 2024.
[90] Llama, "The Llama 3 Herd of Models," 2024.
[91] OpenAI, "GPT-4o System Card," 2024.

Zhongsu Luo received his B.S. and M.S. degree in software engineering from Zhejiang University of Technology. He is currently pursuing a doctoral degree with the State Key Lab of CAD&CG, Zhejiang University. His research interests include data wrangling, human computer interaction, and visualization.

Di Weng is a ZJU100 Young Professor at the School of Software Technology, Zhejiang University. His main research interest lies in information visualization and visual analytics, focusing on interactive data transformation and spatiotemporal data analysis. He received his Ph.D. degree in Computer Science from the State Key Lab of CAD&CG, Zhejiang University. Before his current position, Dr. Weng was a researcher at Microsoft Research Asia from 2022 to 2023. For more information, please visit https://dwe.ng.

Jiawen Zhu received the B.S. degree from Ocean University of China in 2024. He is currently pursuing the Ph.D. degree with the School of Software Technology, Zhejiang University. His research interests include data visualization and time series forecasting.

Shuhan Liu received her B.S. degree in computer science from Zhejiang University in 2021. She is currently pursuing a doctoral degree with the State Key Lab of CAD&CG, Zhejiang University. Her research interests include spatiotemporal data mining, visualization, and industrial data visual analytics.

Xiwen Cai is an engineer in Department of Digital Intelligence, China Mobile. His current job is developing GUI agents and related techniques. He received his B.Sc. degree in Psychology, and M.Sc degree and Ph. D. degree in Computer Science from Zhejiang University. His past research interests include human computer interaction and intelligent data analysis techniques.

Ran Chen received his B.Eng. and Ph. D. degree from Zhejiang University. His research interests mainly lie in tools and systems to author visualizations.

Kai Xiong is a Data Science Researcher at the NLP Department of Hithink RoyalFlush Information Network Co., Ltd (THS). He earned his Ph.D. in Computer Science from Zhejiang University and currently serves as a jointly-trained postdoctoral researcher co-sponsored by THS and Zhejiang University. His primary research focuses on visual analytics and data wrangling. He also has a keen research interest in generating high-quality training data through data synthesis, cleaning, and selection to enhance the performance of large language models (LLMs). For more information, please visit https://xkkevin.github.io/

Jiajun Zhu is an undergraduate research intern at the State Key Lab of CAD&CG, Zhejiang University. He is pursuing his B.S. degree in Computer Science at Chu Kochen Honors College, Zhejiang University (expected 2026). His research interests lie in data visualization and human-AI interaction. For more information, please visit https://jiajunzhuchris.github.io/.

Xinhuan Shu is currently a lecturer at Newcastle University. Her research focuses on designing human–AI collaborative systems to democratize how people engage with data through visualization. Prior that, she was a postdoctoral researcher at the University of Edinburgh and received her Ph.D. degree from HKUST. Her personal website is https://shuxinhuan.github.io/.

Yingcai Wu is a Professor at the State Key Lab of CAD&CG, Zhejiang University. His main research interests are information visualization and visual analytics, with focuses on urban computing, sports science, immersive visualization, and social media analysis. He received his Ph.D. degree in Computer Science from The Hong Kong University of Science and Technology. Prior to his current position, Dr. Wu was a postdoctoral researcher in the University of California, Davis, from 2010 to 2012, and a researcher at Microsoft Research Asia from 2012 to 2015. For more information, please visit http://www.ycwu.org.