Rigel: Transforming Tabular Data by Declarative Mapping

Ran Chen, Di Weng, Yanwei Huang, Xinhuan Shu, Jiayi Zhou, Guodao Sun, Yingcai Wu

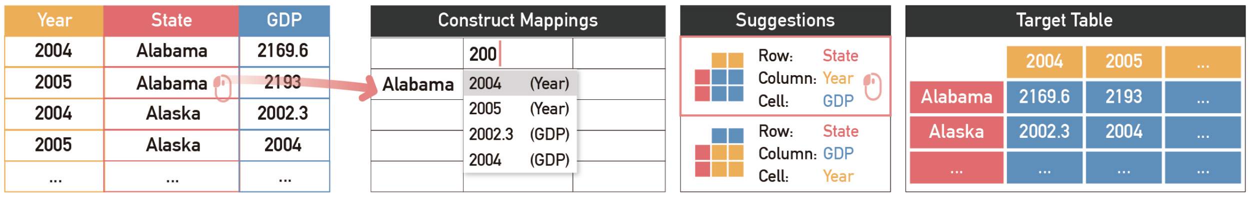


Fig. 1. Rigel Overview. Rigel allows users to construct mappings by dragging or typing values from input data into a spreadsheet. Rigel recommends possible mappings from data to row, column, and cell channels to declaratively compose user-desired tables.

Abstract— We present Rigel, an interactive system for rapid transformation of tabular data. Rigel implements a new declarative mapping approach that formulates the data transformation procedure as direct mappings from data to the row, column, and cell channels of the target table. To construct such mappings, Rigel allows users to directly drag data attributes from input data to these three channels and indirectly drag or type data values in a spreadsheet, and possible mappings that do not contradict these interactions are recommended to achieve efficient and straightforward data transformation. The recommended mappings are generated by enumerating and composing data variables based on the row, column, and cell channels, thereby revealing the possibility of alternative tabular forms and facilitating open-ended exploration in many data transformation scenarios, such as designing tables for presentation. In contrast to existing systems that transform data by composing operations (like transposing and pivoting), Rigel requires less prior knowledge on these operations, and constructing tables from the channels is more efficient and results in less ambiguity than generating operation sequences as done by the traditional by-example approaches. User study results demonstrated that Rigel is significantly less demanding in terms of time and interactions and suits more scenarios compared to the state-of-the-art by-example approach. A gallery of diverse transformation cases is also presented to show the potential of Rigel's expressiveness.

Index Terms—Data transformation, self-service data transformation, programming by example, declarative specification

1 Introduction

Transforming raw data into consumable forms is a critical step in data analysis and decision making [39, 40, 49]. However, the data transformation process is difficult, tedious, and time-consuming, occupying up to 80% of the time and efforts in data analysis [19, 38].

Traditionally, data transformation is performed by editing data record by record, which is apparently not scalable to large datasets. Advanced users resort to writing custom scripts to transform data in batches, preferably leveraging additional data processing libraries [1, 2, 4, 6]. However, learning to write such scripts is challenging due to the time and efforts devoted. In addition, these scripts are mostly tailored for specific datasets and cannot be easily generalized.

Many interactive systems have emerged from recent studies [36, 38, 57] to make data transformation more accessible and efficient to a broader audience. These systems provide various data operations (e.g., aggregating, filtering, and splitting) and recommend appropriate operations for subsequent transformation [25, 38, 65]. However, most recommendations are stepwise and not suitable for complex transformation tasks, which tend to involve lengthy and multiple operations. To address this issue, some systems such as Foofah [36], follow a by-example approach, which synthesizes multi-step operations and recommend transform procedures based on the input example. For instance, a user can change "John Smith" to "Smith, John" in a table cell, obtaining an operation sequence that swaps the first and second strings and adds a comma in between. Although such an approach improves the quality of recommendations, two limitations are observed:

Disambiguating examples. Users are required to provide high-quality examples without ambiguities [24, 43, 68]; otherwise, user intents may be misinterpreted, resulting in incorrect or overfitting suggestions and excessive computational overhead. For instance, the above example can also be interpreted as swapping the first four and the last five characters and then adding a comma in between, which overfits the user intent.

Exploring diverse results. This approach assumes that users have a clear transformation goal in mind. However, in certain scenarios such a goal may not exist [39, 46] and the by-example approach falls short in assisting users in exploring different transformation results to figure out their goals. For instance, data analysts are usually faced with ambiguous requirements such as "what can the dataset be used for?" and "how can the dataset be presented in a readable manner?" They should explore a large number of table forms for insights and find suitable forms for reporting.

In this paper, we introduce Rigel, a general data transformation system that incorporates a new declarative mapping approach, assisting users in addressing the aforementioned limitations. Inspired by the idea that a table is inherently a visual arrangement of data [58, 67], Rigel divides a table into the row, column, and cell channels and implements a novel declarative mapping approach that formulates the transformation procedure as the direct mappings from data to these channels. For instance, Fig. 1 demonstrates a transformation that maps the *State*, *Year*, and *GDP* attributes of the input data into the row, column, and cell channels of the target table, respectively. Hence, each row in the target table will represent a state, each column will represent a year, and each

• *R. Chen, Y. Huang, J. Zhou, Y. Wu are with the State Key Lab of CAD&CG, Zhejiang University, Hangzhou, China. Y. Wu is also with the Alibaba-Zhejiang University Joint Research Institute of Frontier Technologies, Hangzhou, China. E-mail: {chenran928, huangyw, jiayizhou, ycwu}@zju.edu.cn.*
• *D. Weng is with the Microsoft Research Asia, Beijing, China. E-mail: diweng@microsoft.com. D. Weng is the corresponding author.*
• *X. Shu is with the Hong Kong University of Science and Technology, Hong Kong, China. Email: xinhuan.shu@gmail.com.*
• *G. Sun is with the Zhejiang University of Technology, Hangzhou, China. Email: guodao@zjut.edu.cn.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

cell will comprise the GDP for the corresponding state and year. Rigel can transform the input data and compose the desired target table based on such mappings without ambiguities.

Rigel employs an interaction-driven approach, allowing users to directly drag data attributes from input data to the row, column, and cell channels and indirectly drag or type data values in a spreadsheet, and possible mappings are recommended. An example of this approach is illustrated in Fig. 1. In this example, Jonathan drags "*Alabama*" from the input data into the first cell of the second row in the target table (leaving the first row empty for the column header), indicating that he wants to lay the state names down vertically in the first column as the row header. Then, he types "*200*" in the second cell of the first row, and Rigel matches the value he typed with the input data and suggests that year "*2004*" can be filled in this cell. Based on Jonathan's interactions, Rigel can infer and make a transformation recommendation that *State* and *Year* can be mapped into the row and column channels, respectively, and *GDP* will be a reasonable choice for the cell channel. After Jonathan accepts the recommendation, the rest of the target table will be filled automatically and the data transformation is completed. Combining the declarative mapping approach, the interactions, and the recommendations in Rigel, users can feed tables of their interests, intuitively compose the table attributes and values no matter whether their target tables are completely figured out, and explore possible tables to determine the final one to store in the database, analyze in Excel or Tableau, present in reports or visualizations, etc.

We evaluate Rigel with two approaches following the suggestions by Ren et al. [51]. We first assess Rigel's expressiveness with a gallery of diverse transformation cases that highlight Rigel's substantial coverage of transformation tasks [40]. Second, we evaluate Rigel's usability with a quantitative user study, where Rigel is compared against Foofah [35, 36], a by-example data transformation system, across a set of data transformation tasks. By analyzing task performance and qualitative feedback, we find that Rigel is significantly less demanding in terms of time, interactions, and suits more scenarios.

The contributions of this study are summarized as follows:

- We propose a novel declarative mapping approach that formulates the data transformation procedures as mappings from data into the row, column, and cell channels of the target table;
- We develop Rigel, an expressive and user-friendly data transformation system that addresses the disambiguation and exploration issues based on the declarative mapping approach.

2 Related Work

2.1 Foundations of Data Transformation

Many theories, models, and frameworks have been proposed to characterize data transformation. Early data transformation emerges from the longstanding popularity of data query and management on relational data (e.g., SQL and its extensions [10, 41]) in the database community [21, 50]. Consequently, the transformation procedures are often modeled as a sequence of data operations borrowed from the database community. A data operation can be seen as a declarative verb (e.g., *splitting* a column and *filtering* rows) describing *how* to transform the input tables into the desired tables step by step.

With the growing variety of data transformation tasks, additional data operations have been continuously developed. For instance, Wrangler introduces new operations for diverse transformation tasks, such as aggregation, complex reshaping, and conditional mapping [38]. In addition, spreadsheet tools incorporate operations to manipulate strings [22, 54] or the layout [11, 26] of table cells. Moreover, researchers also develop operations that transform the given tables into other data models (e.g., networks [15,32,47]). However, the added data operations also increase the complexity of data transformation.

To address the issue, researchers advocate a *compact* design of data operations, i.e., completing a wide variety of transformation tasks with a limited number of well-designed operations, most of which accept identical parameters [25, 39]. Kasica et al. [40] characterize the operations with two dimensions, namely, three types of data object (*row*, *column*, and *table*) and five categories of operation (*create*, *delete*, *transform*, *separate*, and *combine*). As such, most operations could be identified by the combination of a data type and an operation category. For instance, the operation dividing a table into multiple subsets is depicted as *separating* a *table*. Such a framework presents a clear organization of diverse operations and their parameters, potentially smoothing the learning curve for these operations. Furthermore, it can also improve the efficiency of transformation recommendation models by outlining a boundary for the search space [25].

Although efforts have been devoted to compact data operations, the transformation procedures that comprise operation sequences may still get lengthy, and constructing them become laborious for users as the transformation tasks grow complex [22, 36]. Therefore, we employ a declarative mapping approach and formulate the data transformation procedure as direct mappings from data to the row, column, and cell channels of the target table. These mappings declaratively describe *what* the transformed data will look like in a "squeezed" and bounded structure, which are more systematic, straightforward, and interpretable compared to procedural operation sequences.

2.2 Transformation Systems for Tabular Data

The transformation systems tailored for tabular data have been developed in both industry (e.g., Tableau Prep Builder [5], Trifacta [7], and OpenRefine [3]) and academia (e.g., Flashfill [22], Foofah [36], and Falx [57]). The most common approach that these systems employ to assist users in data transformation is to present data operations in a menu, where users can select and apply the operations to data. Some systems further enable users to preform specific data operations with interactive data visualizations [37, 44, 62, 63]. For instance, brushing in the bar chart *filters* data records out of the scope in Trifacta. Nevertheless, a large set of complex data operations, such as *folding* and *unfolding*, are difficult to perform with interactions. To support diverse data operations, the existing systems either allow users to write custom scripts or configure the parameters for built-in operations with widgets. Despite the expressiveness, writing scripts or configuring operations usually bring steep learning curves and are time-consuming. To balance the expressiveness and usability, modern transformation systems employ recommendation models to suggest appropriate operations. These models can be divided into two categories: single- and multi-step recommendations [66].

Single-step recommendation. Some data transformation systems like Trifacta [7] and its predecessor Wrangler [25, 38] provide single-step suggestions of data operations by user interactions or proactively. Specifically, users can highlight the data features of interest, and these highlighted features will guide the recommendation of data operations. This is called *predictive interaction* [31]. For instance, the system would recommend deleting the empty row or deleting all empty rows in the table if the user clicks on an empty row. Additionally, some systems also proactively analyze data features to recommend single-step operations prior to users' interactions [25, 65]. Although such single-step recommendations can accelerate data transformation, the iterative recommend-and-select processes may get lengthy and laborious for users in difficult tasks [36]. In addition, users are required to understand the recommended data operations, which introduce extra cognitive load and usability issues, especially for non-expert users [11, 36, 38, 42].

Multi-step recommendation. Motivated by the limitations, some other systems follow the by-example paradigm [45] and recommend the sequences of data operations based on the examples provided by the users. For instance, given ("*John Smith*" $\rightarrow$ "*Smith, John*") as an example, a multi-step recommendation system would suggest a sequence of two operations, where the first and last names are exchanged first, and then a comma is added in between. The by-example recommendation of the operation sequence is straightforward, allowing users to focus on their desired output while decoupling tedious execution details from the process. Therefore, it is widely applied to various transformation systems with multiple rounds of iterations and improvements [8, 9, 11, 20, 22, 26, 28, 36, 50, 54–57]. However, two limitations are observed in these systems. First, the key to a successful transformation relies on users to provide high-quality examples without ambiguities [24, 43, 68]; otherwise, the recommendation engine cannot precisely capture user intents and may provide incorrect suggestions with extra computational overhead. To address this issue, some systems move beyond the by-example approach and explore new possibilities. Auto-Type [64] and Auto-Transform [29] systems proactively analyze data features to offer multi-step suggestions without examples. Auto-Pipeline offers recommendation based on the existing transformation pipelines rather than user-provided examples [66]. Nevertheless, these systems still have trouble in understanding user intents due to the lack of sufficient user input. Second, in certain scenarios, such as finding suitable table forms for presentations, users do not have clear transformation goals [39, 46]. The existing systems fall short in assisting users in exploring different transformation results and figuring out user goals.

To conclude, there lacks a transformation system of tabular data that can sufficiently understand user intents in data transformation and facilitate the exploration of diverse transformations when the goals are unclear. Therefore, we propose Rigel to address the limitations observed in the current approaches, employing a novel declarative mapping approach to effectively characterize user intents and recommend diverse target tables based on partial examples.

3 Declarative Mapping for Transformation

To keep the benefits and address the limitations of the transforming by example approaches, we follow the insights of these approaches where users tend to directly specify what the transformed results look like [22, 23], but develop a new approach to formulate user intents into unambiguous table examples. Specifically, a table is inherently a visual arrangement of interrelated data in a row-column form [58, 67]. Based on this idea, we divide a table into three channels including row, column, and cell and model the data transformation procedure as the specification of the transformed target table using the mappings from input data to describe what each row, column, or cell represents:

$$table := (row), (column) \rightarrow (cell), \tag{1}$$

$$row \mid column \mid cell := var \tag{2}$$

Such mappings act like a bridge between user intents and target tables: these mappings can be formulated from explicit or implicit user interactions, and subsequently the target tables can be unambiguously composed from these mappings. To design a solid mapping-based approach, we drew inspiration from the visual mapping in the visualization community that maps data variables to graphical elements (e.g., circle and bar) for visualizations [14, 48] and its successful applications [17, 52]. Researchers also investigate the partial specification of visual mappings for open-ended exploration [60, 61], which inspires us to extend our approach to help users figure out their goals.

To obtain and apply such mappings, we first need to build data variables and their relations from the input data (Fig. 2 (A) and (B)), and then transform the input data into the target table based on the specified mappings (Fig. 2 (C)).

3.1 Building Data Variables and Relations

The declarative mapping approach needs to build a set of data variables and their relations from the input data for users to compose their desired tables. In this study, we use relational tables, which are widely used in many database management systems, as the input data. Other types of input data may also be supported with tailored extraction methods, but we will leave this as future work as it is beyond the scope of this study.

Each column of a relational table can be regarded as a data variable. The possible values for a data variable can be obtained by enumerating unique values in the corresponding column. The values of the data variables in the same row are linked together to construct the relations among the data variables. For instance, as shown in Fig. 2 (A), six data variables are extracted from two relational tables. The "2004" of the *Year* data variable is linked to the "Alabama" and "Alaska" of the *State* data variable, and these two states are subsequently linked to the "2169.6" and "2002.3" of the *GDP* data variable.

A set of functions can be used in the mappings to obtain new data variables from the existing ones as follows:

$$var := f(var) \tag{3}$$

Inspired by the taxonomy of data wrangling operations proposed by Kasica et al. [40], we categorize these functions into three categories:

- **Transform** (1:1). The transform functions can construct a new variable from an existing one. Some of these functions construct new variables to transform each row in a column into different formats, thereby keeping the same numbers of values as the original one. Others are to separate or combine the rows in a column, thus increasing or decreasing the number of values. For example, the *unnest* function converts the arrays or sets in a row into multiple rows. By contrast, aggregating functions (e.g., *bin* and *sum*) summarize multiple values of rows into one. A special case is *filtering*, which removes rows instead of separating or combining.
- **Separate** (1:N). The separate functions can create several variables from the original one. A typical separate function is *splitting*, which splits a string variable by a given separator into a list of variables. The desired variable in this list can be selected with a subscript enclosed in the brackets. For example, `Split(Name, ' ')[0]` maps the first names from the *Name* variable. These functions are mainly for separating columns.
- **Combine** (N:1). The combine functions can produce a variable from multiple variables. One use case is to combine multiple columns, such as *concatenating* the city, street, and building number into a new string describing the address. Another case is to combine multiple tables, such as calculating the *union* or *intersection* of the related variables in different tables to associate these tables for subsequent mappings.

We design these functions by borrowing several principles from the declarative programming paradigm, including *immutability* (data variables can only be derived but not mutated), *purity* (calling these functions does not have side effects and always leads to the same result if the same parameters are supplied), and *composability* (functions can be composed to achieve complex mappings). As such, these functions can be used in the table mappings defined by Equation 1 to declaratively describe the complex transformations from the input relational tables to the target tables. The substantial difference between these functions and traditional data transformation operations is that these functions are to derive new variables instead of transforming tables. In particular, they only process one-dimensional variables comprising a set of data values instead of two-dimensional row-column data tables. Moreover, we exclude the *create* (0:1) and *delete* (1:0) categories proposed by Kasica et al. [40] from the declarative mapping approach because the functions in these categories violate the purpose of the functions to derive new variables, namely, both the input and output cannot be zero. Nevertheless, similar functionalities are implemented in the user interface, where users can add new variables by loading new datasets and delete variables via interactions.

Fig. 2 (B) shows an example of applying these functions to data variables to derive new ones for further transformation. In this example, `Split` is applied to the *State_Abbr* variable to extract the abbreviations of the states from this variable. `Union` merges two identical *Year* variables and their relations, equivalent to performing the join operation for two input relational tables on the *Year* column. `Format` transforms the *GDP* variable into a more readable format. We will show how to compose these variables into a table mapping that describes the transformation process in the next section.

3.2 Transforming Data with Declarative Mappings

The data variables can be mapped into table channels to produce the desired table. For example, the target table in Fig. 2 (C) can be formulated with the following mapping:

$$(\texttt{Split}(State\_Abbr, ...)[1]), (\texttt{Union}(Year, Year)) \rightarrow (\texttt{Format}(GDP, ...)),$$

where each row represents the abbreviation of a state, each year represents a year, and each cell will comprise the GDP formatted in thousands for the corresponding state and year.

To execute transformation based on such mappings, the proposed approach enumerates the values of the variables mapped to the row and column channels and fills these values from the starting cells of rows and columns. Next, the approach searches the values of the variables

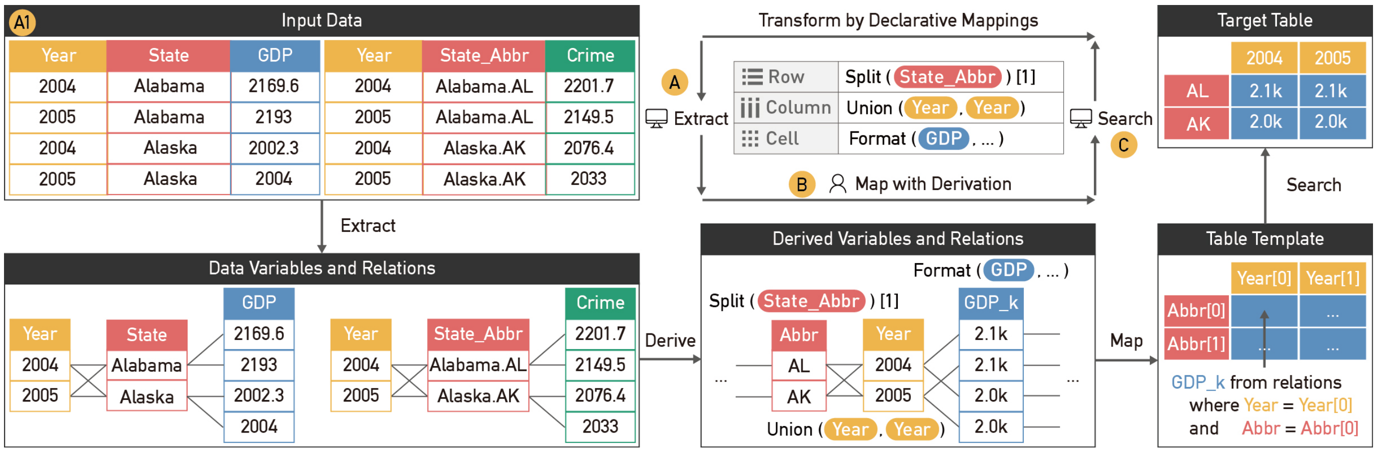


Fig. 2. Rigel's declarative mapping approach for transforming tabular data. (A) The approach extracts data variables and their relations by scanning the input data. (B) Users can map data variables to row, column, and cell channels with data operations to compose their desired tables. (C) Relations between variables search and place data values into corresponding cells to produce the target tables.

mapped to the cell channel and places them into the corresponding cells based on the relations of these variables to the variables mapped to the row and column channels.

If multiple variables need to be mapped into the same table channel, these variables can be composed based on the following rules:

$$var := var \times var \tag{4}$$

$$var := var + var \tag{5}$$

Equation 4 is applied in the row and column channels to generate the Cartesian products of multiple variables in these channels and enumerate tuples in rows and columns. For instance, the relational table in Fig. 2 (A1) can be specified as $(Year \times State), () \rightarrow (GDP)$, where both *Year* and *State* are mapped to the row channel such that each row is associated with a state and a year, and *GDP* is mapped to the cell channel to describe the GDP of a state in a year. In addition, Equation 5 is applied in the cell channel to concatenate different data variables to present multi-dimensional data. For instance, $(Year \times State), () \rightarrow (GDP + Crime)$ can append *Crime* as a new column to the relational table in Fig. 2 (A1), such that the GDP and crime rate of a state in a year are shown in the same row.

Based on the above definition, the following two conditions can be used to check the validity of a table specification:

1. Either the *row* or *column* is not empty;
2. Equation 5 can not be applied in cross-tabulations where both the *row* and *column* are not empty.

Combining functions and mappings of data variables, the approach could cover most of multi-table data wrangling operations (20/21) based on a state-of-the-art taxonomy [40], and we present the details in the example gallery at `https://observablehq.com/@rigel/appendix`.

4 Rigel

This section presents the design considerations and elaborates the system design of Rigel.

4.1 Design Considerations

The design of Rigel should meet the user requirements of various transformation tasks with high usability in both target-oriented transformation and open-ended exploration. To this end, we further instantiate a graphical user interface based on the proposed declarative mapping approach. The design of Rigel is guided by the following considerations, which are based on existing theories of interactive interfaces [16,33,34], spreadsheets [12], exploratory search [30,59–61], and refined during multiple rounds of design iterations.

DC1. Support flexible construction of mappings. *Can the novice users effortlessly create mappings to express their intents? Can the experienced users rapidly and accurately transform data by mapping?* The expertise of users who need to transform data varies from using generic tools to writing custom scripts. Rigel should empower diverse users to perform transformation tasks with different ways according to their levels of expertise, trading off between usability and flexibility.

DC2. Fix ambiguities and conflicts in constructing mappings. *Can Rigel detect ambiguous interactions which could be interpreted in multiple ways, as well as conflict examples which may fail in transformation? Can Rigel help users resolve these issues?* Users may face challenges due to flexible and diverse interactions. To construct mappings, Rigel should detect ambiguities and conflicts from user interactions and assist them in expressing and refining their intents accurately.

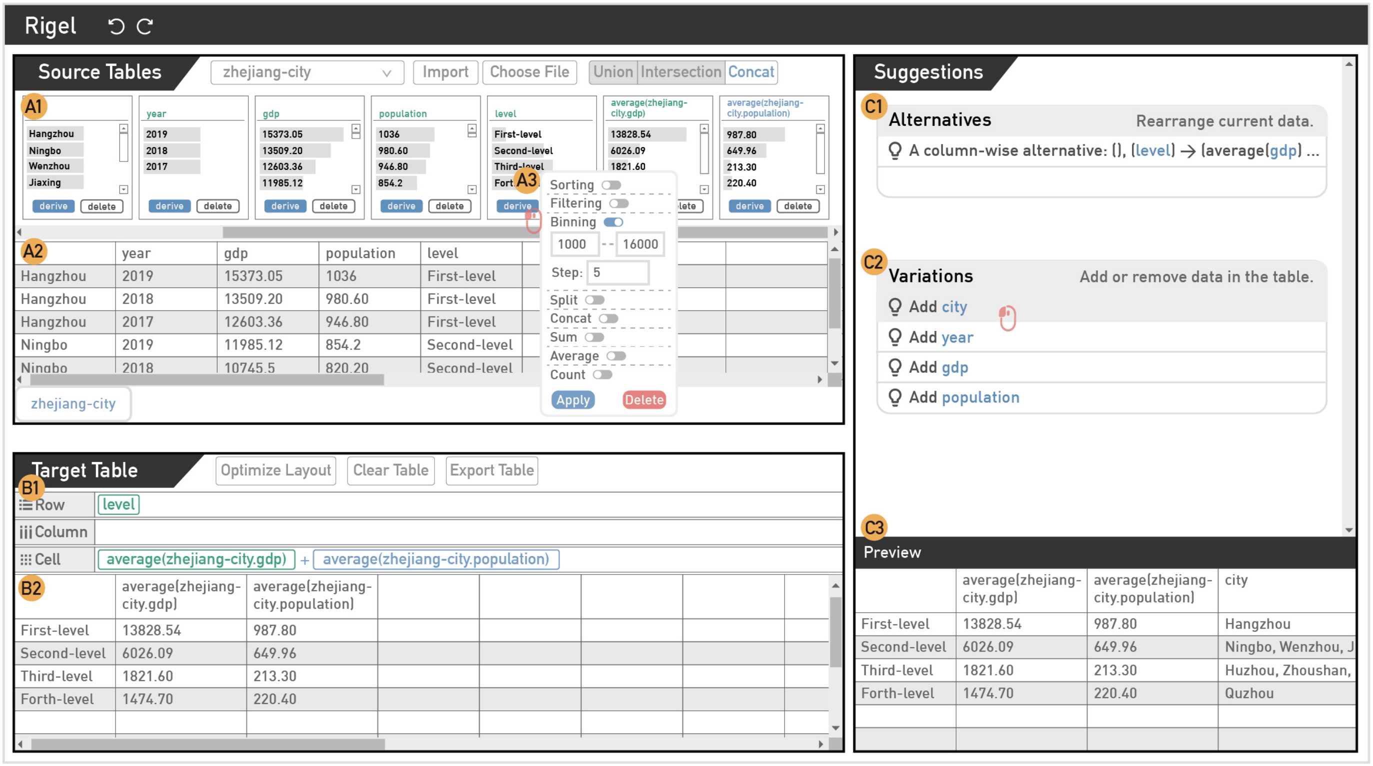
DC3. Recommend for complete and incomplete mappings. *Can Rigel assist users in the exploration of transformations if the users do not have clear goals? Will Rigel help users discover better target tables than what the users have in their mind?* Rigel should employ a mixed-initiative approach to help users find target tables that complete ambiguous tasks or complete tasks better. Such an approach should support inferring potential transformations when no input is given and also accepting partial mappings as constraints for recommendation.

DC4. Balance diversity and exhaustiveness of recommendations. *Can Rigel recommend diverse target tables? Are similar tables pruned to avoid overwhelming users? Does the recommendation satisfy users' need?* There are numerous ways to transform the input data. In particular, many transformations will result in similar tables. Due to limited screen space, Rigel should avoid exhaustively enumerating similar tables which may overwhelm users, but offer diverse recommendations that facilitate the exploration of the target table space.

DC5. Facilitate understanding and selecting of recommendations. *Can Rigel help users read many recommendations? How does Rigel facilitate informed selection among them?* Despite the balance between diversity and exhaustiveness (DC4), various recommendations may still be generated to facilitate the open-ended exploration of transformations. To help users understand these recommendations and choose one of them judiciously, Rigel should aggregate recommendations into distinct categories, present recommendations in readable formats, and preview the recommended tables to facilitate exploration.

4.2 The Rigel System

The Rigel system consists of three views, i.e., a raw-table view to present the raw data tables, a target-table view to compose the resulting table, and a suggestion view to recommend possible transformations with previews, as shown in Fig. 3. Based on the declarative mapping approach as described in Sect. 3, users' pipeline in Rigel is divided into a two-phase iterative workflow as follows:



	year	gdp	population	level
Hangzhou	2019	15373.05	1036	First-level
Hangzhou	2018	13509.20	980.60	First-level
Hangzhou	2017	12603.36	946.80	First-level
Ningbo	2019	11985.12	854.2	Second-level
Ningbo	2018	10745.5	820.20	Second-level

Binning: 1000 -- 16000, Step: 5

Row: level
Cell: average(zhejiang-city.gdp) + average(zhejiang-city.population)

	average(zhejiang-city.gdp)	average(zhejiang-city.population)
First-level	13828.54	987.80
Second-level	6026.09	649.96
Third-level	1821.60	213.30
Forth-level	1474.70	220.40

Alternatives — Rearrange current data.
A column-wise alternative: (), (level) → (average(gdp) ...

Variations — Add or remove data in the table.
Add city
Add year
Add gdp
Add population

	average(zhejiang-city.gdp)	average(zhejiang-city.population)	city
First-level	13828.54	987.80	Hangzhou
Second-level	6026.09	649.96	Ningbo, Wenzhou, J
Third-level	1821.60	213.30	Huzhou, Zhoushan,
Forth-level	1474.70	220.40	Quzhou

Fig. 3. Rigel's user interface. (A) Raw-table view presents (A1) data variables and (A2) their relations of the raw data. (A3) Users can click the "derive" button and apply data functions to derive new variables. (B) Target-table view allows users to compose target tables by (B1) directly mapping variables into three table channels or (B2) indirectly assembling examples for mappings in a spreadsheet. (C) Suggestion view recommends (C1) alternative tables by rearranging the current table and (C2) table variations by adding unused data variables for open-ended exploration, and (C3) supports hover to preview the effects.

1. *Construct table mappings.* Rigel allows users to compose tables by constructing declarative mappings. In particular, users can directly map variables into table channels or indirectly assemble table examples for mappings.
2. *Explore table variants.* With the current table, Rigel generates related variants for recommendation. Users can preview and choose the most suitable one, or they can skip and return to the first phase to provide more hints for transformation.

The two phases are iterated until users have obtained satisfactory tables. Additional interactions are also integrated in this workflow to provide some basic functionalities similar to those implemented in general spreadsheet tools, such as *undo*/*redo* and *import*/*export*.

4.2.1 Construct Table Mappings

The variables (Fig. 3 (A1)) and their relations (Fig. 3 (A2)) of the imported data are extracted and depicted in the raw-table view. Users can map these variables from the raw-table view to the target-table view to compose target tables for data transformation. Rigel provides two classes of interactions to construct mapping (DC1): *direct* mapping and *indirect* mapping. Users can further apply functions for mappings and should resolve ambiguities and conflicts during the process. Each interaction produces a partial mapping from a variable to a table channel, and Rigel merges these partial mappings from multiple interactions to create a complete mapping and render the resulting table.

The *direct* mapping enables users to directly drag a variable from the raw-table view (Fig. 3 (A1)) to one of three table channels in the target-table view (Fig. 3 (B1)), thereby generating a partial mapping.

The *indirect* mapping allows users to indirectly construct mappings by assembling examples in the spreadsheet of the target-table view (Fig. 3 (B2)) and select recommendations in the suggestion view (Fig. 3 (C)). Users can assemble examples in three ways (DC1). First, users can drag a data value from a cell in the raw-table view (Fig. 3 (A2)) into a cell in the target view. Second, users can also drag a data variable (a set of data values) in the raw-table view (Fig. 3 (A1)) into cells in the target-table view. Dragging the variable into the left border of a cell means placing the variable values into cells of the same row started by the cell. Similarly, dragging into the top border of a cell means placing the values into cells of the same column started by the cell. Third, Rigel allows users to directly type text into cells in the target-table view with the assistance of autocomplete.

However, the indirect mapping may suggest multiple possible mappings (DC3) and Rigel should detect ambiguities for users to determine the desired mapping. For instance, as shown in (Fig. 4), dragging the value "First-level" into a cell of the target table implies mapping the variable *level* to the row or column channel. Based on the declarative mapping approach, the ambiguities of the indirect mapping may originate from two aspects: the variable and the table channel (DC2). First, the assembled example could match multiple variables and users have to determine the variable for mapping. In particular, such ambiguities are mainly due to typing instead of dragging which implicitly indicates the variable. Second, Rigel enumerates possible table channels for the variable based on the distribution of the variable along rows or columns. Specifically, Rigel recommends the row or column channel if examples are filled row-wise or column-wise. The cell channel recommendation follows the two validity conditions of mappings proposed in Sect. 3.2 in addition to the distribution: (1) if either the row or column channel is not empty, or (2) if both the row and column channels are not empty and the cell channel is empty. For instance, when the row channel is non-empty and the column channel is empty, Rigel will recommend the cell channel if the examples are distributed row-wise but will not recommend if the examples are column-wise. During the two steps of disambiguation, all suggestions are presented in natural language based on predefined templates like Fig. 4 (DC5).

During the direct and indirect mappings, users can apply functions to derive new variables in both raw- and target-table views (DC1). In

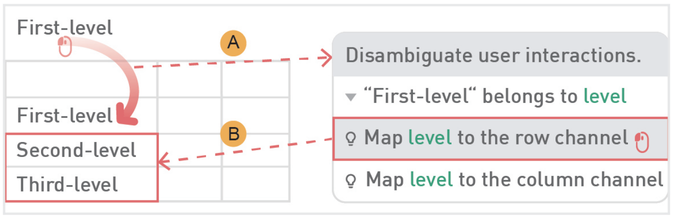

First-level
A
Disambiguate user interactions.
"First-level" belongs to level
First-level
B
Second-level
Map level to the row channel
Third-level
Map level to the column channel

Fig. 4. (A) Users can assemble examples by dragging data into the target table, and (B) Rigel composes the target table by providing possible mappings for users to disambiguate the examples.

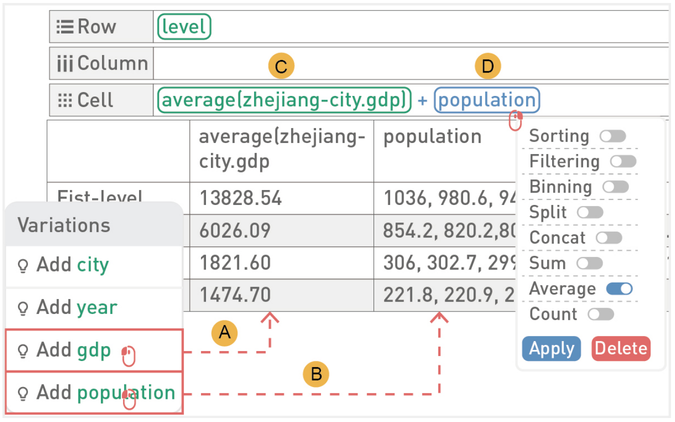


Fig. 5. (A, B) Users can extend the current table with the recommended table variations. (C, D) Users can also fine-tune the current table by right clicking variables and applying functions.

the raw-table view, users can click the "derive" button of a variable to transform or separate one variable to one or multiple new variables (Fig. 3 (A3)). Users can also drag a variable to another one to combine the variables into a new variable based on the selected combine function (Union/Intersection/Concat). In the target-table view, users can transform and separate variables by right clicking (Fig. 5 (D)) and combine variables by dragging one to another in the table channels. In particular, applying functions to variables in the target-table view is a shortcut to deriving variables in the raw-table view and then replacing the original variables using the derived ones in the target table.

The constructed partial mappings are merged into the existing mappings in the target-table view (Fig. 3 (B1), and the resulting table in the spreadsheet is re-rendered (Fig. 3 (B2)). Specifically, variables in the same table channel of different partial mappings are composed based on Equation 4 and Equation 5. For instance, $(A),() \rightarrow (B)$ and $(),() \rightarrow (C)$ will be merged into $(A),() \rightarrow (B+C)$. The merged mapping can be modified by adding, moving, and deleting variables (DC1). Moreover, due to the overly flexible interactions of indirect mapping, there may exist unreasonable layout of the mapping. For instance, there may exist conflicts that different partial mappings share the same cell. In addition, mappings may also distribute far away from each other, resulting in a loose layout. To address the issues (DC2), Rigel enables users to click the "Optimize Layout" button to replace the current layout with a standard layout of the mapping.

4.2.2 Exploring Table Variants

Based on the constructed mapping and the resulting table, Rigel recommends related table variants to inspire users to explore and choose the most suitable one. Similar as the visualization recommendation of visual mappings in Voyager [60, 61], Rigel receives all variables in tables and the current table mapping, which may be incomplete or complete during user interactions, and outputs two sets of related table mappings: *variations* and *alternatives* (DC3 and 5).

We denote the set of variables in the current mapping as V_p and the set of all variables of the used tables as V. $V_q = V - V_p$ denotes the set of unused variables in these tables. For instance, for the variables and the mapping in Fig. 3 (A1) and (B1), V_p has three variables: *level*, *average(gdp)*, and *average(population)*, and V_q has the other four variables related to the table *zhejiang-city*: *city*, *year*, *gdp*, and *population*.

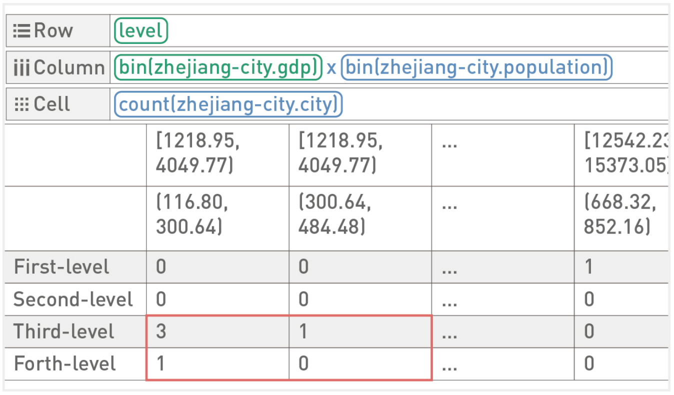
Variations are mappings that add unused but related variables (V_q) to the current table mapping to enrich the current table. With variations, extra variables can be added to mappings to iteratively extend small and simple tables to large and complex tables. When significant variables are all selected, *alternatives* are helpful to rearrange the variables in the current table mapping (V_p) into different channels for different table forms. Enumerative approaches can exhaustively search all combinations of variations and alternatives. However, the search space is huge and can produce overwhelming recommendations. In particular, assuming n and m variables of V_p and V_q, $(n+3)m$ variations can be recommended by enumerating each of the m variables in $n+3$ different positions in the three table channels of the current mapping. In addition, there exist more than $n!$ alternatives by permuting the n variables in table channels. Nevertheless, the enumerative recommendations usually produce similar tables, most of which cannot inform users of diverse table forms but cause heavy cognitive load. Therefore, Rigel only recommends typical variations and alternatives (DC4), encouraging users to manually select the one that is close to their target tables and fine-tune for the tables by dragging and deriving variables in table channels. Specifically, Rigel lists one variation for an additional variable in V_q and maps the variable into the end of the cell channel (Fig. 3 (C2) and (C3). Besides, Rigel enumerates three typical types of alternatives, namely, table mappings without *row*, without *column*, and cross-tabulation with *row*, *column*, *cell*, and recommends the three types if they are valid for the current mapping (Fig. 3 (C1)). All recommendations are presented in natural language using predefined templates and support hover to preview the effect (DC5).

5 Usage Scenario

To demonstrate the expressiveness and usability of Rigel, we walk through a data analyst Jane's process of exploring the economy and population of different levels of cities in Zhejiang province, China. The input dataset comprises 33 rows and 5 variables (*city*, *year*, *gdp*, *population*, and *level*) (Fig. 3 (A2)), where each row describes a city's GDP, population, and classification of tier levels by media publications in a year. The dataset is finally transformed into a cross-tabulation to reveal the distribution of cities of different tier levels in different GDP and population ranges, as shown in Fig. 7.

Jane gets this table for the first time and does not have a clear goal for data analysis. She starts her exploration by first dragging the "First-level" in the raw table to the target table out of her interest in different city levels (Fig. 4 (A)). Rigel then derives two possible transformations in the suggestion view that infer the user intent, i.e., map the variable *level* to the row or column channel. After previewing the effects applied these suggestions respectively via hovering, Jane selects $(level),() \rightarrow ()$ as a trial (Fig. 4 (B)). Rigel updates the recommendation accordingly, providing alternatives and variants based on the current table. Jane applies her interested variables *gdp* and *population* sequentially (Fig. 5 (A) and (B)). As such, Rigel automatically maps *gdp* and *population* into the cell channel, i.e., each row represents a certain level with two sets of *gdp* and *population* values belonging to this level. Jane then right clicks *gdp* and *population* in the target table to compute the average of both sets by applying the *average* operation (Fig. 5 (C) and (D)). As such, she gets a table (Fig. 3 (B2)) showing the average *gdp* and *population* values in different city levels.

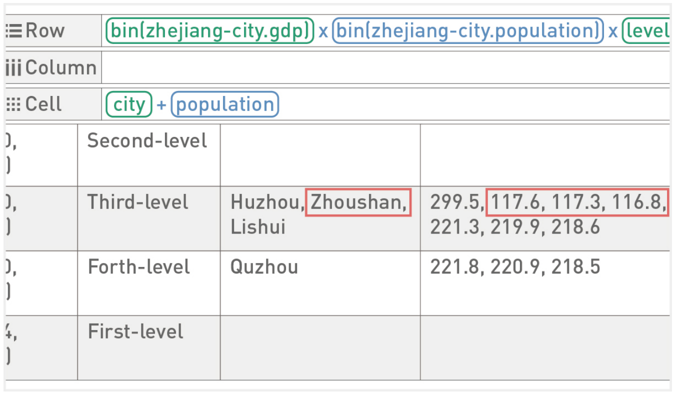
Intrigued by two data patterns in the table: synchronous drop between the city level and GDP, population, and abnormal rise of population from the third- to fourth-level cities, Jane determines to further check the distribution of cities in different GDP and population ranges. She clicks the "Derive" button and applies the default bin to *gdp* and *population*. Next, she removes *average(gdp)*, *average(population)* from the target table, and drags the binned *gdp*, *population* into the column channel directly. Then, Jane drags *city* into the cell channel



Row	level
Column	bin(zhejiang-city.gdp) x bin(zhejiang-city.population)
Cell	count(zhejiang-city.city)

	[1218.95, 4049.77)	[1218.95, 4049.77)	...	[12542.23 15373.05
	(116.80, 300.64)	(300.64, 484.48)	...	(668.32, 852.16)
First-level	0	0	...	1
Second-level	0	0	...	0
Third-level	3	1	...	0
Forth-level	1	0	...	0

Fig. 6. $(level), (\texttt{bin}(gdp) \times \texttt{bin}(population)) \rightarrow (\texttt{count}(city))$ specifies a cross-tabulation presenting the distribution of different levels of cities in a series of range combinations of *gdp* and *population*.



Row	bin(zhejiang-city.gdp) x bin(zhejiang-city.population) x level
Column	
Cell	city + population

),)	Second-level		
),)	Third-level	Huzhou, Zhoushan, Lishui	299.5, 117.6, 117.3, 116.8, 221.3, 219.9, 218.6
),)	Forth-level	Quzhou	221.8, 220.9, 218.5
4,)	First-level		

Fig. 7. $(\texttt{bin}(gdp) \times \texttt{bin}(population) \times level), () \rightarrow (city + population)$ reshapes the table in Fig. 6 and complements population values, showing that "Zhoushan" has an extremely low population during three years.

and applies *count*. The table (Fig. 6) shows the distribution of different levels of cities in a series of range combinations of *gdp* and *population*. Inspecting the table, Jane notices that the first- and second-level cities are distributed in high and middle GDP and population ranges respectively, while both third- and fourth-level cities in low ranges, validating synchronous drop between the city level and GDP, population, and indicating the gap between different levels of cities. However, the abnormal rise of population from the third- to fourth-level cities still cannot be understood as the third-level cities are distributed in the lowest and penultimate low population ranges and the fourth-level cities are all distributed in the lowest range, which also follows the pattern of synchronous drop. To investigate this anomaly, Jane directly adds *population* to the cell channel for details of the distribution and folds the cross-tabulation to obey Equation 5 for a valid table. The resulting table (Fig. 7) reveals that a third-level city "Zhoushan" has extremely low populations, pulling down the overall average.

With Rigel, Jane can rapidly transform the input dataset into different formats, facilitating data analysis and exploration and gaining insights.

6 User Study

To assess Rigel's usability, we conducted a task-based user study to compare Rigel and Foofah [35, 36], a by-example data transformation system. We hypothesized that Rigel would be efficient and require less efforts to complete transformation tasks.

6.1 Method

The user study employs a 2 (data transformation system) $\times$ 4 (task) mixed design. Each participant conducted two data transformation sessions, each with a different data transformation system to perform the same four data transformation tasks. We randomized the order of systems and tasks across subjects. The behaviors and subjective feedback of subjects were collected and analyzed to verify our hypothesis and demonstrate Rigel's usability.

Participants. We recruited 12 participants (6 males and 6 females), all students (5 PhD, 3 MS, and 4 undergraduate) from different departments, including Computer Science (6), Electronic Engineering (1), Media (1), Art (1), Agronomy (1), Psychology (1), and Sport Science (1). The participants are represented as P1–P12 in this paper. All subjects were with prior data transformation experience (average 3.9 years and self-reported expertise 3.08 on a 5 point Likert scale). Subjects had used data transformation tools including Microsoft Excel, MATLAB, SPSS, GraphPad, Python, R, and none of them had used Rigel, Foofah, or similar by-example transformation systems before.

Data Transformation Systems. Participants interacted with two data transformation systems: Rigel and Foofah, one of the state-of-the-art systems for transforming tabular data by example. Foofah supports common transformation tasks, such as dropping, splitting, and transposing, and incorporates a by-example approach such that users provide input-output examples for Foofah to synthesize programs to transform the entire input data. Compared to other by-example transformation systems targeting programming users, such as AutoPandas [13] and Wrex [20], Foofah is mainly for average users, providing smooth interactions without needs of inspecting program scripts, which is fairly appropriate to compare with Rigel.

Tasks and Data. Participants are required to complete four data transformation tasks using both Foofah and Rigel. The design of the four tasks is informed by common practices of data workers according to our experience and is based on the expressiveness of Foofah and Rigel. In particular, Foofah incorporates operators to rearrange and reshape tables, group rows, manipulate strings, and incorporate missing values, but lacks operators to combine and separate tables, such as joining and aggregating. Meanwhile, Rigel is designed to transform tables and not excel at manipulating strings and interpolating missing values. Therefore, the four tasks mainly involve grouping and filtering rows, string manipulation, and table reshaping. To reduce cognitive load, we use datasets with at most 4 data variables and 20 records. In each task, subjects should transform a raw data table into a new format, as described below:

1. *Aggregate Observations.* Subjects started with data containing papers and authors in a relational format. They should group data records by paper names, aggregate paper authors into a cell, and separate author names with commas.
2. *Generate Unique Key.* We gave subjects a relational table of housing crime data by years and states and asked them to generate a unique key for each row by concatenating *year* and *state*.
3. *Create Cross-tabulation.* With a table comprising people's phone and fax numbers at home and work, subjects should remove incomplete records and create a cross-tabulation of numbers by people and number types.
4. *Transpose Table.* Subjects were required to transpose the same table of housing crime in Task 2 from 20 rows $\times$ 3 columns into 3 rows $\times$ 20 columns.

Study Protocol. Each data transformation session began with a 10-minute tutorial to walk through the system using a dataset distinct from those used in subsequent tasks. Next, the subjects freely explored to get familiar with the system and could ask us for unclear features for 5 minutes. Afterward, they performed the four data transformation tasks following the think-aloud protocol (permitting a maximum of 5 minutes per task), and their behaviors and resulting tables were recorded. During the process, we did not respond to their inquires. After completing tasks, the subjects were asked to fill in a post-study questionnaire designed using a 7-point Likert scale based on NASA-TLX [27]. Finally, we interviewed subjects with several open-ended questions, e.g., ranking system features, discussing confusion, and sharing feedback. The entire process lasted approximately 1.5 hours, and we compensated each subject with a $15 gift coupon.

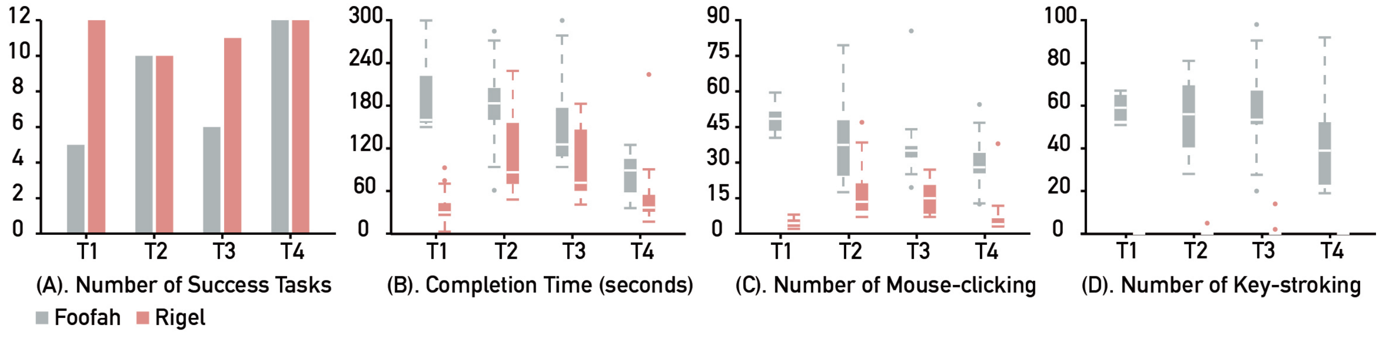


Fig. 8. Results of four data transformation tasks using Foofah and Rigel by users. In general, Rigel completed (A) more tasks and was significant less demanding in terms of (B) time and interactions including (C) mouse clicks and (D) keystrokes.

Evaluation Metrics. To quantify the amount of user efforts on both systems, we measured the time and the number of mouse clicks and keystrokes of each task. Combining these three metrics helps reduce negative impacts of other factors, such as using the thinking aloud protocol in the study. Furthermore, we counted the number of different ways subjects used to assemble examples and construct mappings in Rigel to understand user preference to these interactions.

6.2 Quantitative Results

Subjects quickly learned how to transform data in both Foofah and Rigel. However, they encountered difficulties during tasks. As shown in Fig. 8 (A), about half of subjects failed in Task 1 and 3 with Foofah, and a small number of users could not complete Task 2 (15 failures of 48 attempts in total). These failures mainly originated from overly simple examples users provided that could not sufficiently convey their intents, or overly complex examples that are difficult for Foofah to understand. In contrast to Foofah, there exist fewer failed tasks in Rigel (3/48). These failures resulted from user's unfamiliarity with the by-mapping approach, including the *combine* function (Equation 3) in Task 1 and the *Cartesian products* (Equation 4) in Task 3.

Across the successful tasks, subjects completed the tasks with less time, mouse clicks, and keystrokes in Rigel than Foofah (Fig. 8 (B–D)). Specifically, subjects usually started Foofah by providing output examples with about five to ten cells by selecting, copying, and pasting or directly typing values. If the first round of examples failed to complete the tasks, Foofah only presents the wrong transformed results, which usually confused subjects and helped little to understand mistakes. Subjects tended to randomly feed more examples to Foofah, leading to laborious trial-and-error attempts (average 2.8, 2.2, 1.8, and 1 round of examples for the four tasks respectively). Moreover, the synthesizer of Foofah also took time to search data transformation programs based on examples. By contrast, Rigel incorporates flexible dragging interactions and efficient recommendation, significant reducing efforts in assembling examples. All subjects liked dragging and almost abandoned the interactions of text typing, selecting, copying, and pasting in Foofah. Only three subjects typed examples into cell once or twice during their tasks. As such, few keystrokes were recorded in the study. In addition, subjects preferred the direct mapping of variables in table channels to the indirect mapping of examples in the spreadsheet. Therefore, significantly fewer mouse clicks for dragging were required.

6.3 Qualitative Feedback

Subjects' qualitative rating and comments reinforce the quantitative results. In particular, all subjects agreed that the by-example and by-mapping transformation approaches suit different usage scenarios. We summarize the following insights to discuss their differences from the aspect of human cognition and perception.

6.3.1 Example and Mapping: Execution v.s Evaluation

Examples and mappings are two types of direct manipulation interfaces [34] of data transformation and have different characteristics of directness for users. All subjects agreed that Foofah's by-example approach is straightforward and almost has no learning threshold. A subject said that "*I even did not have to think but just conducted rounds of trial-and-error of examples to complete tasks*." In contrast to examples, although most subjects (8/12) thought mappings are accessible, they also acknowledged the existence of a learning curve. P3 started with reversed understanding of the row channel and the column channel in mappings. Specifically, P3 usually subconsciously treated the row and column channels as table headers along the row and column direction. For instance, for a mapping $(A),() \rightarrow ()$, P3 thought that Rigel would enumerate and fill the values of A in cells of a row, which is the result of $(),(A) \rightarrow ()$. As such, P3 spent too much time with a large number of interactions on the attempt of the first task Task 4, resulting in the outliers of T4 in Fig. 8 (B) and (C). Nevertheless, P3 corrected the misunderstanding of the declarative mapping after multiple rounds of attempt and rapidly completed the other three tasks. Finally, P3 also appreciated the high interpretability of mappings and commented "*the status of the table-to-date can be checked at any time so I can figure out my fault rapidly*.", which is difficult in examples where subjects had to inefficiently guess the effects and randomly assemble examples. About half of subjects (7/12) felt doubtful and unconfident of their examples in Foofah ($\mu = 4.08, \sigma = 3.08$ on a 7-point Likert scale of the frustration level). One subject commented "*I felt like the example is sufficient, but had no idea why the result is wrong*." Another subject were irritated to multiple rounds of trial-and-error and gave up one task.

Despite the benefits of mappings, most subjects did not like constructing mappings by examples and commented the way "*stressful*" because they have to determine the partial mapping of examples immediately. They hope that they focus on assembling examples with optional mapping recommendations, and Rigel finally recommends the complete mapping and table. Apart from the mapping, several subjects talked about the shortcomings of the derivation functions. For instance, P2 and P3 forgot the combine function of string concatenation and failed in Task 2. P2 pointed out "*the functions are not put together*", which informs us to unify the interactions of the same type of functionalities to keep the consistency and facilitate the accessibility. In particular, the combine functions should also be placed in the function list triggered by the "derive" button, and the dragging interaction for the combine functions can be a shortcut.

In summary, **mappings have a slightly larger *gulf of execution* [34] than examples while having a considerably narrower *gulf of evaluation* [34] as a trade-off.** In particular, as said by P3, "*Rigel will be expressive and efficient if users could understand Rigel's approach*." By contrast, examples are suited for those who occasionally need to complete simple or similar data transformation tasks (6/12 subjects). P7 mentioned the by-example approach may facilitate a typical scenario of "*bringing new data into existing data*" for novice users, such as transforming the sales data into the same format as the last month. To narrow the gulf of execution, tuning Rigel's user interface to approach existing tools with additional shortcuts is helpful. Moreover, Rigel would benefit from a new interaction model that breaks the *premature commitment* [16] of examples and mappings and enables users to focus more on assembling examples.

6.3.2 Example and Mapping: Details v.s Overall

Examples and mappings are usually with different qualitative feeling of engagement for different users. Although most users commented examples and mappings were "*natural*", there exist counterexamples. P1 and P4 mentioned that examples did not fit their logic and brought in extra thinking overhead. P4 said that she usually figured out overall shapes and formats of target tables, but "*would not consider the specific value of each cell*." P1 pointed out that he tended to offer transformation logic to systems and viewed results, and thought the detailed result assembly with examples "*unnatural*". Both P1 and P4 guessed such thinking logic may originate from the programming logic since they major in Computer Science. By contrast, P2 who majors in Art and usually designs from local to global liked the by-example approach and thought that examples provided fine-grained control for customizing individual needs. To conclude, **mappings are suited for those who like to transform data from the overall logic of tables while examples support fine-grained control of details.**

6.3.3 Mapping: Systematic Exploration

P3, P4, and P5 mentioned that **enumerating mappings helped open-ended exploration.** P3 said that he did not fully understand the by-mapping approach (as described in Sect. 6.3.1) but could also completed data transformation tasks. P3 pointed out that the mapping provided a enumerable, systematic template to organize tables, and similar tables were with similar mappings via short edit distances. Therefore, he performed efficient trial-and-error exploration of various tables through independent changes of variables, table channels, and gradually approached target tables of tasks. During the exploration, P3 said that he also discovered some useful tables he did not expect.

7 Discussion

Implications. This study proposes to integrate a declarative mapping approach into a by-example data transformation approach. From the perspective of techniques, Rigel implements a result-oriented data transformation approach grounded in a novel table definition by declarative mapping that abstracts traditional process-oriented approaches. Rigel incorporates the approach into an interactive data transformation system for example disambiguation and open-ended exploration. From the perspective of evaluation, this study analyzes several patterns of usability for data transformation from the aspects of human cognition and perception, which could provide insights for designers of data transformation toolkits and guide the development of these toolkits. From the perspective of usefulness, Rigel helps data workers improve their efficiency significantly. The declarative mapping approach can also be implemented in a textual environment as a general data transformation library. From the perspective of generalizability, the table definition proposed by Rigel covers various table forms and has the potential to be extended to a unified characterization of data tables. First, it depicts the organizations of data in tables and provides a succinct overview of tabular data. Second, it allows to specify tables and facilitates the search from search engines or other corpus (e.g., Google Dataset Search [18]). Third, it can provide entries to analyze real-world tables (e.g., clustering table specifications), which can promote novel and creative table designs, efficient exploration of the table space, and advanced models for intelligent table generation and recommendation.

Lessons learned. We present two design lessons we have learned while developing Rigel. First, data transformation can also be regarded as a process of visual information seeking. Therefore, data transformation tools should also follow the mantra "overview first, then details-on-demand" [53] to offer usable user interface. During iterations of Rigel, we developed an alternative design that separates variable derivations and mapping specifications for conciseness of both steps, instead of nesting them in a table specification. However, the feedback was disappointing because users preferred to start transformations with mappings to characterize the overall shape of data tables and then operate data values on demand. With the alternative design, users had to switch the two steps mentally, which increased their burdens. Second, analyzing system design from cognitive dimensions of notations [16] is useful. Specifically, each round of Rigel's design was analyzed from these dimensions and was accordingly refined to improve the usability. For instance, targeting high *closeness of mapping*, we leveraged user familiar objects (data variables, rows, columns, and cells) to design Rigel's mapping, removing abstract concepts (e.g., entity) and improving accessibility.

Limitations and future work. This work explores to apply an overall declarative mapping approach for data transformation from the underlying model to the interactive system. However, the current implementation of Rigel is admittedly modest, which focuses more on supporting the breadth-oriented feasibility of a taxonomy of wrangling operations [40] instead of exhausting all possible operations. We observe the limitations in both the declarative mapping approach and the interactive system, and discuss the potential future directions.

Rigel's declarative mapping approach currently incorporates a few basic functions with limited parameters and a simple mapping structure with three common table channels. The gallery has demonstrated its feasibility to handle most types of data operations. Furthermore, the approach can be expanded to handle more specific tasks of each type of the operations. For instance, interpolating missing values based on the above values (*fill* in wrangler [38] or *ffill* in pandas [4]) is capable of Rigel with additional fine-grained mappings between values and cells. One possible implementation is to leverage logical functions, e.g., $(A),() \to (Or(B,\text{"}above\text{"}))$, such that if a value of B is missing then the above value is mapped to the corresponding cell. Another example is reshaping cross-tabulations with hierarchical headers such that the identical values of adjacent header cells are omitted (*stack* in pandas). This task can be realized by incorporating a nested row channel into the existing row channel following Wang's table anatomy [58]. This analysis encourages us to develop more features in engineering for an expressive data transformation approach.

The design of the Rigel system is intentionally limiting the interactions to fixed steps of disambiguating examples and determining partial mappings while restricting the recommendations to interpretable enumerations and additions of single variables. Such a design reduces the complexity. The user study presents Rigel's benefits of reducing interactions, understanding transformations, suiting people who prefer the top-down logic, and helping a systematic exploration. In addition, Rigel also exposes drawbacks of a slightly steep learning curve, the constraints of users' workflow, and the capability of recommending complex tables. Therefore, we regard advanced interaction and recommendation techniques as important future work. A possible solution is allowing users to focus more on assembling examples with optional but not mandatory partial mappings, and finally recommends the complete mapping and table.

8 Conclusion

This paper introduces Rigel, a general data transformation system with a declarative mapping approach to disambiguate user intents and facilitate open-ended exploration. Rigel formulates the data transformation procedure as direct mappings from data to row, column, and cell channels of the target table, and enables users to flexibly construct mappings based on recommendations to compose their target tables. Through a diverse example gallery and a task-based user study, we demonstrate that Rigel supports expressive data transformation tasks and enables efficient data transformation compared to by-example approaches.

Rigel is an open-source system available at `https://github.com/rigel-js/rigel-system`, and the declarative mapping approach is also open-sourced as a JavaScript library at `https://github.com/rigel-js/rigel.js`. We plan to collect real-world feedback and devote more engineering efforts to develop more features for a solid data transformation library and system in the future.

Acknowledgments

The work was supported by NSFC (62072400) and the Collaborative Innovation Center of Artificial Intelligence by MOE and Zhejiang Provincial Government (ZJU). This work was also partially funded by the Zhejiang Lab (2021KE0AC02).

References

[1] Arquero, a javascript library for query processing and transformation of data tables. https://github.com/uwdata/arquero. (Retrieved: Jan 15th, 2022).

[2] dplyr, a grammar of data manipulation. https://dplyr.tidyverse.org. (Retrieved: Jan 15th, 2022).

[3] Open Refine. https://openrefine.org. (Retrieved: Jan 15th, 2022).

[4] pandas, a python data analysis and manipulation tool. https://pandas.pydata.org. (Retrieved: Jan 15th, 2022).

[5] Tableau Prep. https://www.tableau.com/products/prep. (Retrieved: Jan 15th, 2022).

[6] tidyr, tidy messy data. https://tidyr.tidyverse.org. (Retrieved: Jan 15th, 2022).

[7] Trifacta. https://www.trifacta.com. (Retrieved: Jan 15th, 2022).

[8] Z. Abedjan, J. Morcos, M. N. Gubanov, I. F. Ilyas, M. Stonebraker, P. Papotti, and M. Ouzzani. DataXFormer: Leveraging the Web for semantic transformations. In *Proceedings of the Conference on Innovative Data Systems Research (CIDR)*, 2015.

[9] Z. Abedjan, J. Morcos, I. F. Ilyas, M. Ouzzani, P. Papotti, and M. Stonebraker. DataXFormer: A robust transformation discovery system. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*, pages 1134–1145, 2016.

[10] R. Ahmed, P. D. Smedt, W. Du, W. Kent, M. A. Ketabchi, W. A. Litwin, A. Rafii, and M. Shan. The pegasus heterogeneous multidatabase system. *Computer*, 24(12):19–27, 1991.

[11] D. W. Barowy, S. Gulwani, T. Hart, and B. Zorn. FlashRelate: extracting relational data from semi-structured spreadsheets using examples. In *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, pages 218–228, 2015.

[12] L. Bartram, M. Correll, and M. Tory. Untidy data: The unreasonable effectiveness of tables. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 28(1):686–696, 2022.

[13] R. Bavishi, C. Lemieux, R. Fox, K. Sen, and I. Stoica. AutoPandas: neural-backed generators for program synthesis. *Proceedings of the ACM on Programming Languages*, 3(OOPSLA):168:1–168:27, 2019.

[14] J. Bertin. Semiology of graphics: Diagrams, networks, maps. Technical report, 1983.

[15] A. Bigelow, C. Nobre, M. Meyer, and A. Lex. Origraph: Interactive network wrangling. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 81–92, 2019.

[16] A. F. Blackwell, C. Britton, A. L. Cox, T. R. G. Green, C. A. Gurr, G. F. Kadoda, M. Kutar, M. Loomes, C. L. Nehaniv, M. Petre, C. Roast, C. Roe, A. Wong, and R. M. Young. Cognitive dimensions of notations: Design tools for cognitive technology. In *Proceedings of the International Conference on Cognitive Technology: Instruments of Mind*, pages 325–341, 2001.

[17] M. Bostock, V. Ogievetsky, and J. Heer. D^3 data-driven documents. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 17(12):2301–2309, 2011.

[18] D. Brickley, M. Burgess, and N. Noy. Google Dataset Search: Building a search engine for datasets in an open Web ecosystem. In *Proceedings of the World Wide Web Conference (WWW)*, pages 1365–1375, 2019.

[19] T. Dasu and T. Johnson. *Exploratory Data Mining and Data Cleaning*. John Wiley, 2003.

[20] I. Drosos, T. Barik, P. J. Guo, R. DeLine, and S. Gulwani. Wrex: A unified programming-by-example interaction for synthesizing readable code for data scientists. In *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems (CHI)*, pages 1–12, 2020.

[21] H. Galhardas, D. Florescu, D. E. Shasha, and E. Simon. AJAX: an extensible data cleaning tool. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, page 590, 2000.

[22] S. Gulwani. Automating string processing in spreadsheets using input-output examples. In *Proceedings of the ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)*, pages 317–330, 2011.

[23] S. Gulwani, W. R. Harris, and R. Singh. Spreadsheet data manipulation using examples. *Communications of the ACM*, 55(8):97–105, 2012.

[24] S. Gulwani, O. Polozov, and R. Singh. Program synthesis. *Foundations and Trends® in Programming Languages*, 4(1-2):1–119, 2017.

[25] P. J. Guo, S. Kandel, J. M. Hellerstein, and J. Heer. Proactive wrangling: Mixed-initiative end-user programming of data transformation scripts. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*, pages 65–74, 2011.

[26] W. R. Harris and S. Gulwani. Spreadsheet table transformations from examples. In *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, pages 317–328, 2011.

[27] S. G. Hart. NASA-task load index (NASA-TLX); 20 years later. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 50, pages 904–908. Sage Publications Sage CA: Los Angeles, CA, 2006.

[28] Y. He, X. Chu, K. Ganjam, Y. Zheng, V. R. Narasayya, and S. Chaudhuri. Transform-Data-by-Example (TDE): an extensible search engine for data transformations. *Proceedings of the VLDB Endowment*, 11(10):1165–1177, 2018.

[29] Y. He, Z. Jin, and S. Chaudhuri. Auto-Transform: Learning-to-transform by patterns. *Proceedings of the VLDB Endowment*, 13(11):2368–2381, 2020.

[30] M. Hearst. *Search user interfaces*. Cambridge University Press, 2009.

[31] J. Heer, J. M. Hellerstein, and S. Kandel. Predictive interaction for data transformation. In *Proceedings of the Conference on Innovative Data Systems Research (CIDR)*, 2015.

[32] J. Heer and A. Perer. Orion: A system for modeling, transformation and visualization of multidimensional heterogeneous networks. *Information Visualization*, 13(2):111–133, 2014.

[33] E. Horvitz. Principles of mixed-initiative user interfaces. In *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems (CHI)*, pages 159–166, 1999.

[34] E. L. Hutchins, J. D. Hollan, and D. A. Norman. Direct manipulation interfaces. *Human-Computer Interaction*, 1(4):311–338, 1985.

[35] Z. Jin, M. R. Anderson, M. J. Cafarella, and H. V. Jagadish. Foofah: A programming-by-example system for synthesizing data transformation programs. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 1607–1610, 2017.

[36] Z. Jin, M. R. Anderson, M. J. Cafarella, and H. V. Jagadish. Foofah: Transforming data by example. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 683–698, 2017.

[37] S. Kandel, J. Heer, C. Plaisant, J. Kennedy, F. van Ham, N. H. Riche, C. Weaver, B. Lee, D. Brodbeck, and P. Buono. Research directions in data wrangling: Visualizations and transformations for usable and credible data. *Information Visualization*, 10(4):271–288, 2011.

[38] S. Kandel, A. Paepcke, J. M. Hellerstein, and J. Heer. Wrangler: Interactive visual specification of data transformation scripts. In *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems (CHI)*, pages 3363–3372, 2011.

[39] S. Kandel, A. Paepcke, J. M. Hellerstein, and J. Heer. Enterprise data analysis and visualization: An interview study. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 18(12):2917–2926, 2012.

[40] S. Kasica, C. Berret, and T. Munzner. Table Scraps: An actionable framework for multi-table data wrangling from an artifact study of computational journalism. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 27(2):957–966, 2021.

[41] L. V. S. Lakshmanan, F. Sadri, and S. N. Subramanian. SchemaSQL: An extension to SQL for multidatabase interoperability. *ACM Transactions on Database Systems*, 26(4):476–519, 2001.

[42] T. Lau. Why programming-by-demonstration systems fail: Lessons learned for usable AI. *AI Magazine*, 30(4):65–67, 2009.

[43] T. Y. Lee, C. Dugan, and B. B. Bederson. Towards understanding human mistakes of programming by example: An online user study. In *Proceedings of the International Conference on Intelligent User Interfaces (IUI)*, pages 257–261, 2017.

[44] G. Li, R. Li, Z. Wang, H. C. Liu, M. Lu, and G. Wang. Hitailor: Interactive transformation and visualization for hierarchical tabular data. *IEEE Transactions on Visualization and Computer Graphics*, To Appear.

[45] H. Lieberman. *Your wish is my command: Programming by example*. Morgan Kaufmann, 2001.

[46] J. Liu, N. Boukhelifa, and J. R. Eagan. Understanding the role of alternatives in data analysis practices. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 26(1):66–76, 2020.

[47] Z. Liu, S. B. Navathe, and J. T. Stasko. Network-based visual analysis of tabular data. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 41–50, 2011.

[48] J. D. Mackinlay. Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics (TOG)*, 5(2):110–

141, 1986.

[49] M. Muller, I. Lange, D. Wang, D. Piorkowski, J. Tsay, Q. V. Liao, C. Dugan, and T. Erickson. How data science workers work with data: Discovery, capture, curation, design, creation. In *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems (CHI)*, page 126, 2019.

[50] V. Raman and J. M. Hellerstein. Potter's Wheel: An interactive data cleaning system. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 381–390, 2001.

[51] D. Ren, B. Lee, M. Brehmer, and N. H. Riche. Reflecting on the evaluation of visualization authoring systems: Position paper. In *IEEE Evaluation and Beyond - Methodological Approaches for Visualization*, pages 86–92, 2018.

[52] A. Satyanarayan, D. Moritz, K. Wongsuphasawat, and J. Heer. Vega-Lite: A grammar of interactive graphics. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 23(1):341–350, 2017.

[53] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of the IEEE Symposium on Visual Languages*, pages 336–343. IEEE Computer Society, 1996.

[54] R. Singh and S. Gulwani. Learning semantic string transformations from examples. *Proceedings of the VLDB Endowment*, 5(8):740–751, 2012.

[55] R. Singh and S. Gulwani. Synthesizing number transformations from input-output examples. In *Proceedings of the International Conference on Computer-Aided Verification (CAV)*, volume 7358, pages 634–651, 2012.

[56] R. Singh and S. Gulwani. Transforming spreadsheet data types using examples. In *Proceedings of the ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)*, pages 343–356, 2016.

[57] C. Wang, Y. Feng, R. Bodík, I. Dillig, A. Cheung, and A. J. Ko. Falx: Synthesis-powered visualization authoring. In *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems (CHI)*, pages 106:1–106:15, 2021.

[58] X. Wang. *Tabular Abstraction, Editing, and Formatting*. PhD thesis, University of Waterloo, Ontario, Canada, 1996.

[59] R. W. White and R. A. Roth. *Exploratory Search: Beyond the Query-Response Paradigm*. Synthesis Lectures on Information Concepts, Retrieval, and Services. Morgan & Claypool Publishers, 2009.

[60] K. Wongsuphasawat, D. Moritz, A. Anand, J. D. Mackinlay, B. Howe, and J. Heer. Voyager: Exploratory analysis via faceted browsing of visualization recommendations. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 22(1):649–658, 2016.

[61] K. Wongsuphasawat, Z. Qu, D. Moritz, R. Chang, F. Ouk, A. Anand, J. D. Mackinlay, B. Howe, and J. Heer. Voyager 2: Augmenting visual analysis with partial view specifications. In *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems (CHI)*, pages 2648–2659, 2017.

[62] K. Xiong, S. Fu, G. Ding, Z. Luo, R. Yu, W. Chen, H. Bao, and Y. Wu. Visualizing the scripts of data wrangling with SOMNUS. *IEEE Transactions on Visualization and Computer Graphics*, To Appear.

[63] K. Xiong, Z. Luo, S. Fu, Y. Wang, M. Xu, and Y. Wu. Revealing the semantics of data wrangling scripts with COMANTICS. *IEEE Transactions on Visualization and Computer Graphics*, To Appear.

[64] C. Yan and Y. He. Auto-Type: Synthesizing type-detection logic for rich semantic data types using open-source code. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 35–50, 2018.

[65] C. Yan and Y. He. Auto-Suggest: Learning-to-recommend data preparation steps using data science notebooks. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 1539–1554, 2020.

[66] J. Yang, Y. He, and S. Chaudhuri. Auto-Pipeline: Synthesize data pipelines by-target using reinforcement learning and search. *Proceedings of the VLDB Endowment*, 14(11):2563–2575, 2021.

[67] R. Zanibbi, D. Blostein, and J. R. Cordy. A survey of table recognition. *International Journal on Document Analysis and Recognition*, 7(1):1–16, 2004.

[68] T. Zhang, Z. Chen, Y. Zhu, P. Vaithilingam, X. Wang, and E. L. Glassman. Interpretable program synthesis. In *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems (CHI)*, pages 105:1–105:16, 2021.