

Taggle: Scalable Visualization of Tabular Data through Aggregation

Katarina Furmanova*, Samuel Gratzl*, Holger Stitz, Thomas Zichner, Miroslava Jaresova, Martin Ennemoser, Alexander Lex, and Marc Streit

Abstract—Visualization of tabular data—for both presentation and exploration purposes—is a well-researched area. Although effective visual presentations of complex tables are supported by various plotting libraries, creating such tables is a tedious process and requires scripting skills. In contrast, interactive table visualizations that are designed for exploration purposes either operate at the level of individual rows, where large parts of the table are accessible only via scrolling, or provide a high-level overview that often lacks context-preserving drill-down capabilities. In this work we present Taggle, a novel visualization technique for exploring and presenting large and complex tables that are composed of individual columns of categorical or numerical data and homogeneous matrices. The key contribution of Taggle is the hierarchical aggregation of data subsets, for which the user can also choose suitable visual representations. The aggregation strategy is complemented by the ability to sort hierarchically such that groups of items can be flexibly defined by combining categorical stratifications and by rich data selection and filtering capabilities. We demonstrate the usefulness of Taggle for interactive analysis and presentation of complex genomics data for the purpose of drug discovery.

Index Terms—Information visualization, visualization techniques and methodologies, tabular data, multi-dimensional data, aggregation, hierarchical grouping and sorting.

1 INTRODUCTION

VISUALIZATION of tabular or multi-dimensional data is important in many application domains and is a mainstay of visualization research. We distinguish three types of tabular data visualization techniques: (1) *overview techniques*, which position marks for each cell based on the cell’s value (for example, parallel coordinates, scatterplot matrices, and generalizations such as *FLINA* [1]); (2) *projection techniques*, which show a lower-dimensional projection of a high-dimensional dataset (such as scatterplots of the output of principle component analysis or multidimensional scaling [2]); and (3) *tabular techniques*, which retain a tabular layout and encode the data within the cells (such as heatmaps and the *Table Lens* [3]). There are also hybrid approaches and systems that use multiple coordinated views to combine the strengths of individual techniques.

While there is a rich body of work on overview and projection techniques, there is less work on tabular techniques, with some notable exceptions, such as the *Table Lens* [3], *Bertifier* [4], *LineUp* [5], and *ComplexHeatmap* [6]. In this paper, we revisit tabular visualization approaches by introducing various generalizations and improvements that make them more flexible and scalable.

Our primary contribution is *Taggle*, a tabular visualization method that allows explicit visualization of large tabular datasets

by selectively aggregating subsets of a dataset. When used with larger data sets, tabular data visualization typically requires a trade-off between overview and detail: for showing an overview, techniques such as the *Table Lens* down-scale rows to be as thin as a single pixel, whereas multiple pixels per row are necessary when labels are to be readable. However, even down-scaling rows has its limits, as information loss occurs when a dataset has more items than the screen has pixels.

The goal of *Taggle* is to provide a high-level overview of large and heterogeneous tabular datasets while allowing users to drill down to individual items. To this end, we use selective aggregation of rows and columns. Aggregation is enabled by various grouping strategies, which can be based on combinations of categorical attributes, user-driven selections, or on setting thresholds to numerical attributes. Both aggregates and individual rows can be visualized using a multiform approach, where users can choose the appropriate visualization techniques for the scale, aggregation status, and type of individual columns. The grouping and aggregation capabilities are complemented by sorting and filtering techniques. The resulting technique is suitable for both interactive exploration and presentation of complex tabular datasets.

We demonstrate *Taggle*’s utility by means of a case study on analyzing a genomic cancer dataset for the purpose of drug discovery. Further, we demonstrate *Taggle* using a public health dataset: the spread of AIDS across the nations of the world.

2 TABULAR DATA

Throughout this paper, we use an AIDS dataset from UNAIDS AIDSinfo¹ as a guiding example. This dataset was enriched with metadata about the countries, such as population, which we retrieved from the United Nations Population Division² and

1. <http://aidsinfo.unaids.org/>

2. <http://www.un.org/en/development/desa/population/>

- K. Furmanova is with Masaryk University Brno.
E-mail: furmanova@mail.muni.cz
- S. Gratzl is with datavisyn GmbH and Johannes Kepler University Linz.
E-mail: samuel.gratzl@datavisyn.io.
- M. Jaresova is with Masaryk University Brno and Czechitas, z.s.
E-mail: mirka@czechitas.cz.
- H. Stitz, M. Ennemoser, and M. Streit are with Johannes Kepler University Linz. E-mail: {holger.stitz, martin.ennemoser, marc.streit}@jku.at.
- T. Zichner is with Boehringer Ingelheim RCV GmbH & Co KG.
E-mail: thomas.zichner@boehringer-ingelheim.com.
- A. Lex is with the University of Utah.
E-mail: alex@sci.utah.edu.
- * These authors contributed equally to this work.

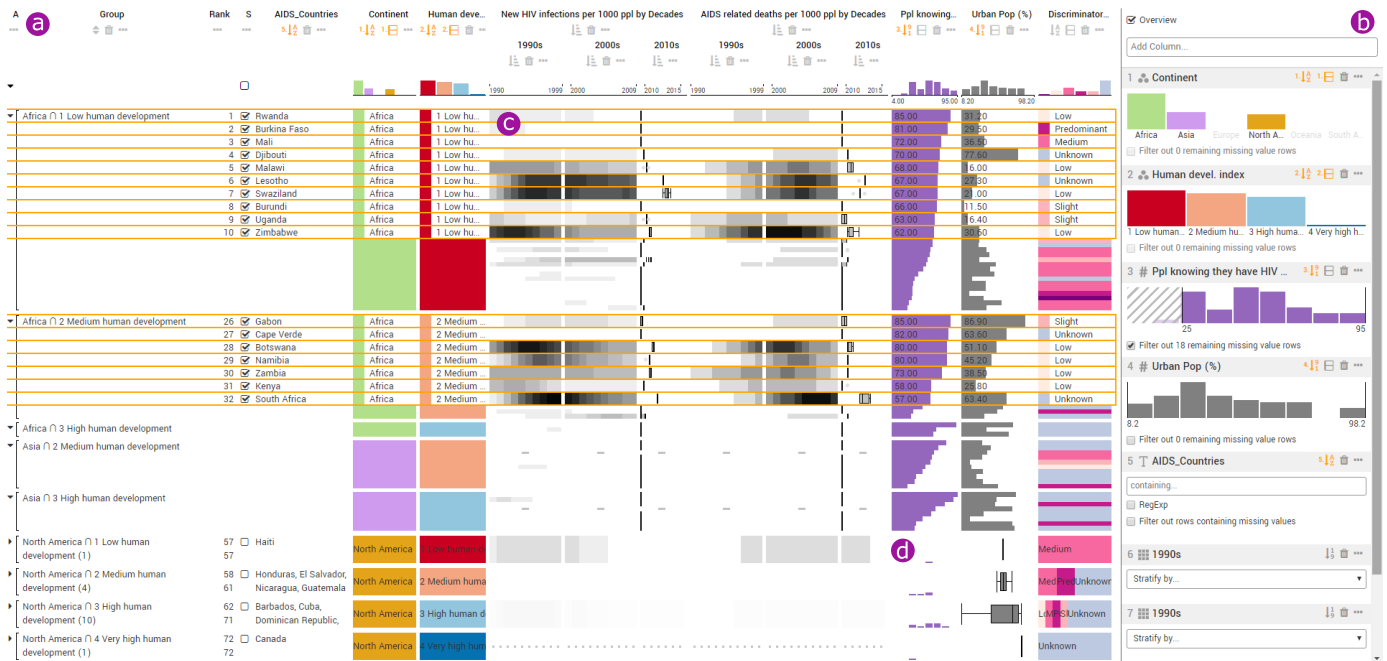


Fig. 1: The Taggle interface consisting of a table view (a) and a data selection panel (b) showing a dataset on AIDS in several countries grouped by continent and level of human development index. The data selection panel consists of attribute filter views that allow users to filter out records by interacting with the histograms. The selected rows indicate the relationship between *new HIV infections* and *AIDS related deaths* in African countries over time. It can be seen that an outburst of *new HIV infections* in the 1990s in southern African countries resulted in high *AIDS-related death* rates about a decade later in the 2000s (c). The rows of countries in *North America* have been aggregated to histograms, box plots, and stacked bars (d).

the yearly Human Development Report of the United Nations Development Programme³. The combined dataset consists of 17 numerical vectors (e.g., *population*, *sex before the age of 15* in percent), 4 categorical vectors (e.g., *continent*, *human development index*), and 10 time-series matrices (e.g., *AIDS related deaths* or *new HIV infections over a period of 27 years*) collected for 160 countries.

Tabular datasets are usually composed of items stored in rows, which often correspond to independent variables (countries, in our example), and values (i.e., observations about these variables) stored in columns, which commonly correspond to dependent variables (e.g., *population*, in our example). Lex et al. [7] discuss heterogeneity and sources of heterogeneity in tabular data: *semantics*—the columns in the table have different meanings; *characteristics*—the columns have different data types and value ranges; and *statistics*—the columns have different behaviors or distributions.

Homogeneous datasets lend themselves to compact and simple visual representations, as all data items share the same meaning and scales. Heatmaps [8], for example, are well suited to homogeneous datasets, as they encode each cell with a color value, which makes it possible to represent individual items at minimal scale.

Heterogeneous datasets have different semantics, characteristics, and statistics. Consequently, they may need separate scales and visual representations for each column. For instance, the *population* is given in absolute numbers and *sex before 15* is stated in percent.

Following this, we distinguish between these data types: **Vectors** are columns where all associated records are of the same type and semantics, such as the *name*, *gender*, and *age* columns in a table of people. Attributes can be categorical, numerical, or

textual. **Matrices** are composed of vectors of the same semantics and data type. An example is a country's *GDP* over multiple years, where each year is a vector in the matrix. While it is possible to interpret matrices as a list of vectors, it is beneficial to treat them as a matrix, because the homogeneity of the data is an opportunity for aggregation. The columns in matrices can also be associated with vectors that describe a common property of the column, such as the decade associated with a year.

3 REQUIREMENTS

Based on discussions with experts from the biomedical domain who regularly analyze and present large tabular datasets, literature reviews, and our own experience, we elicited a set of requirements that an effective scalable, interactive table visualization technique should support. Gratzl et al. [5] and Perin et al. [4] also discuss requirements for tabular data visualization. While some requirements overlap with ours, Taggle is broader in scope and therefore necessitates a broader set of requirements.

R1 Encoding. The table visualization should be able to flexibly encode the data subset contained in the cell, including special encoding for missing data. The content of a cell can be either a single data value (string or number) or multiple data values in the case of matrix data. In addition, the visualization should support various representations for cells that summarize the values of multiple rows (see requirement R5). Examples of such aggregate visualizations are histograms and box plots.

R2 Sorting. An effective table visualization should allow users to sort items by categorical or numerical vectors or by weighted combinations of thereof. It should be possible to form a sorting hierarchy where ties in the first-level attribute are

3. <http://hdr.undp.org/>

broken by the second-level attribute, and so on. Furthermore, users should be able to sort the table based on groups, such as their median, average, or minimum/maximum value (R5).

- R3 Filtering.** The table should provide a rich set of filtering capabilities. Users should be able to filter out one or multiple categories from categorical vectors, set thresholds for numerical vectors, and set textual filters.
- R4 Grouping.** Users should be able to stratify (group) both rows and columns by categorical vectors. The categories can be either predefined categorical attributes, categories that are algorithmically derived (e.g., through binning or clustering) or categories defined by the user through interaction.
- R5 Aggregating.** Aggregating data subsets (i.e., groups of items) is key to making the table scalable to a large number of rows and columns. Aggregation can be achieved by reducing the dimensionality by one: a matrix can be transformed to a vector, and a numerical vector can be reduced to a single value. However, data aggregation introduces uncertainty, which needs to be communicated to the user by choosing suitable encodings (see requirement R1).
- R6 Multiform.** The requirements for data representation may differ based on the user’s task, the level of detail, or the character of the data. The table visualization should therefore provide flexible means for interactively changing the visual encoding of numerical, nominal, ordinal, textual, and matrix data.
- R7 Combining columns.** Giving users the ability to visually combine multiple columns is another key feature a table visualization should support. The most common approach to combining multiple numerical columns is to let users create weighted combinations, which can be represented using, for example, stacked bars. However, for other comparison tasks, different visual encodings are needed. For instance, when comparing two values from datasets collected at different times, or when comparing the interdependency between a categorical variable (e.g., *continent*) and a numerical variable (e.g., *population*), showing two columns juxtaposed can give users an overview of global patterns, but is not conducive to comparing individual values. Being able to interleave or superimpose values of multiple columns in a cell on-demand enables users to make such a detailed comparison. In a cancer dataset, for example, users want to relate the average gene expression of normal tissue to that of cancerous tissue by vertically stacking the corresponding bars that encode the two conditions.
- R8 Transforming data.** Users should be able to transform the data flexibly by setting minimum/maximum values, by clipping the values to a defined range, or by setting the scale. This requirement also affects encoding (R1) where users want to map the input values to the visual channels used for visualizing the columns (e.g., mapping categorical or numerical attributes to colors, or numerical values to the length of the bars).
- R9 Bidirectional matrix operations.** Adding a matrix to a table visualization introduces a second key for the columns of the matrix. It should be possible to sort, group, filter, and aggregate columns of matrix data based on numerical, categorical, and textual vectors.
- R10 Interactive refinement and visual feedback.** In interactive visualization techniques, user actions should immediately be reflected in the visualization. For example, if users dynamically change a filter, items should be removed or added as the filter is modified. This enables users to learn about the data by interacting with it. Furthermore, animated transitions [9] should help users to understand the operations applied to the table.
- R11 Showing an overview of items.** In order to provide the user with an overview, all items should fit into the available space without aggregation—if possible. Conversely, this means that the number of off-screen items, which are only accessible via scrolling, should be minimized.
- R12 Showing details of items.** It should also be possible to show detailed information about items—labels describing individual table items and their precise values should be visible to provide details about the content of the table. This partially contradicts the previous requirement, as labels require more space and therefore decrease the number of items that can be fit into the table visualization. Finding the right balance between showing overview and detail is therefore of great importance.

4 RELATED WORK

We discuss related work under three different considerations: (1) a review of general tabular data visualization techniques, (2) a discussion of benefits and drawbacks of single vs. multiple views for tabular data visualization, and (3) approaches to aggregation.

4.1 Tabular Data Visualization

Since tabular data analysis plays an important role in many research fields, a substantial body of work exists on visualizing such data. We distinguish between three types of tabular data visualization techniques:

- 1) **overview techniques**, which position marks for each cell based on its value, such as parallel coordinates, star plots, and scatterplot matrices,
- 2) **projection techniques**, which show a lower-dimensional projection of a high-dimensional dataset, and
- 3) **tabular techniques**, which retain item positions across columns and encode the data within the cells.

These types of techniques have different strengths and weaknesses: overview techniques show high-level trends and patterns in the dataset across attributes, projection techniques show a general similarity of items, and tabular techniques emphasize the values of individual items. Tabular techniques can also convey global trends when ordering the table either by sorting or by applying matrix-reordering techniques.

There are also hybrid approaches that combine overview and tabular approaches or overview and projection approaches. In hybrid overview-tabular approaches, the rows are preserved within subsets of the data, but the relationships between subsets are visualized using an overview technique. Examples of this class include *NodeTrix* [10], *VisBricks* [7], *StratomeX* [11], *Domino* [12], and *Furby* [13]. In hybrid overview-projection approaches, selected attributes are plotted on top of a plot of projected data, as in the technique developed by Stahnke et al. [14]. In the following, we limit our discussion to tabular and hybrid techniques.

	Requirements											
	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12
Microsoft Excel [15]	~	✓	✓	✓	×	×	×	✓	×	×	✓	✓
Tableau [16]	✓	✓	✓	✓	✓	✓	~	✓	~	~	✓	✓
InfoZoom [17]	✓	✓	✓	✓	×	×	×	✓	×	✓	✓	✓
Complex Heatmap [6]	✓	✓	~	~	×	~	~	✓	✓	×	✓	✓
Domino [12]	✓	✓	×	✓	✓	✓	×	×	✓	✓	×	×
DataComb [18]	✓	✓	✓	×	×	×	×	×	×	✓	✓	✓
Table Lens [3]	✓	✓	✓	×	×	~	×	×	×	✓	✓	✓
LineUp [5]	✓	✓	✓	×	×	×	~	✓	×	✓	×	✓
Bertifier [4]	✓	✓	×	×	×	✓	×	✓	×	✓	×	✓
Taggle	✓	✓	✓	✓	✓	✓	✓	✓	~	✓	✓	✓

TABLE 1: Overview of which state-of-the-art techniques and tools fulfill the requirements defined in Section 3. R1 – Encoding, R2 – Sorting, R3 – Filtering, R4 – Grouping, R5 – Aggregating, R6 – Multiform, R7 – Combining columns, R8 – Transforming data, R9 – Bidirectional matrix operations, R10 – Interactive refinement and visual feedback, R11 – Showing an overview of items, R12 – Showing details of items.

Existing tabular visualization techniques have different strengths and weaknesses. While some techniques focus on the presentation of static tabular data, such as tailored tables summarizing research findings, others support interactive exploration of the data. Taggle aims to serve both presentation and exploration purposes. Table 1 summarizes the relevant tabular visualization techniques and tools with respect to the requirements defined in Section 3.

Interactive & Exploratory Techniques

Widely used spreadsheet tools, such as *Microsoft Excel* [15], *Google Sheets* [19], and *Apache OpenOffice Calc* [20], typically support tabular operations such as sorting, filtering, and grouping (R2, R3, R4). However, while spreadsheet tools usually support rich charting operations, they provide only limited support for direct visual encoding of cells, using techniques such as conditional formatting (R1, R6). *Tableau* [16] can be used to create tabular visualizations that use a variety of visual encodings. However, while Tableau can support most of our requirements, it requires the manipulation of a complex interface that must be learned.

FOCUS [21] and its successor *InfoZoom* [17], [22] aim to address the drawbacks of typical spreadsheet tools by introducing visual encodings in cells (R1). In FOCUS, data is presented in a table that fits all items into the available screen space (R11). For large datasets, this results in unreadable labels for individual items. To address this problem, users can select items which are then displayed at a readable size. Alternatively, a full-size spreadsheet-like mode ensures readability (R12). In addition to spreadsheet and space-filling modes, InfoZoom also offers an overview mode that shows the distribution of values for individual attributes. In this mode, each attribute row is sorted individually, identical values are grouped together, and cell size indicates the size of the group. Both FOCUS and InfoZoom lack support for matrices and associated operations (R9), and do not allow attributes to be combined (R7). Furthermore, the visual encoding used in these tools is limited and cannot be adjusted (R6).

The Table Lens [3] is probably most closely related to Taggle and inspired its development. It uses visual encodings tailored to different data types to represent values in cells (R1). Rich sorting operations allow users to compare trends between separate attributes (R2) and when applied to categorical variables, to group items (R4). Scalability is achieved by down-scaling rows, and a combination of appropriately chosen visual encodings and lens

techniques ensures readability of trends and individual items. The most important differences to Taggle are that the Table Lens does not support aggregation (R5) and is therefore limited in terms of scalability; it also does not support advanced features, such as bidirectional matrices (R9). *DataComb* [18] is a web-based re-implementation of the Table Lens technique. A similar, yet less sophisticated approach, is implemented in the *Visual Spreadsheet* [23], which comes with the *UCSC Xena Functional Genomics Browser* [24].

While the Table Lens and many other table visualizations allow users to rank a table by a single column, LineUp [5] and *ValueCharts* [25] visualize multi-attribute rankings where users can create weighted combinations of attributes, and represent the result as stacked bars (R2, R7).

Domino [12] is a hybrid tabular/overview technique. It is based on the concept of placing subsets of a dataset on a canvas and choosing a suitable representation for it (R6). Multiple subsets can then be connected to show their relationships in various ways. *Matchmaker* [26], *VisBricks* [7] and *StratomeX* [11], [27] are related hybrid techniques, but are more restricted with respect to the selection of subsets.

Both Matchmaker and VisBricks focus on numerical matrices and facilitate the comparison of different groupings or clusterings. Other techniques, such as the hierarchical cluster explorer [28], *GAP* [29], *PermutMatrix* [30], and *Clustergrammer* [31], focus on visualizing clustered matrices. Taggle, in contrast, supports both numerical matrices and heterogeneous tables in a single visualization.

Presentation-Focused Techniques

Several techniques focus on the presentation of tabular data, for example, to be used in publications, as opposed to interactive techniques designed for exploring large datasets.

Bertifier [4] is a table-visualization technique inspired by Jacques Bertin’s matrix analysis methods. It uses various visual encodings for cells (R1, R6) that can be interactively re-ordered based on similarities between rows and columns. Other features, such as styling options for the table grid, indicate that the technique is intended mainly for presenting small or medium-sized tables. Bertifier does not support aggregation (R5) or grouping (R4) and is limited to numerical data.

There are also several plotting tools and libraries, such as *ggplot2* [32] for R and *matplotlib* [33] for Python, that can be used to create static tabular visualizations using scripting/programming. *ComplexHeatmap* [6] is an R package that allows users to represent a matrix as a heatmap, where the rows and columns can be stratified (R4). Additional categorical and numerical vectors as well as derived data can be attached to any of the four sides of the matrix with flexible encoding possibilities (R1).

By leveraging the unmatched expressiveness of programming languages, these tools and others like it can fulfill all of the requirements, with the notable exception of interactive refinement (R10). However, the expressivity comes at a cost, as the creation of effective representations can be time-consuming and requires scripting skills, and the lack of interactivity hinders exploration.

4.2 Multiple-View Systems

Complementary to the three types of tabular data visualization techniques introduced in the previous section, there is an option of representing (groups of) attributes of a tabular dataset independently of each other in a multiple coordinated view system. Representative systems in this category include *Improvise* [34] and the recent *Keshif* [35]. These systems allow users to choose representations that are suitable for the subset of data represented by a single view, and usually rely on linked highlighting to re-introduce connections. Common configurations of *Keshif*, for example, use a tabular view to identify specific items, but represent other attributes in other views using, for instance, histograms or bar charts.

While MCV systems can leverage visualization techniques that are ideal for certain attributes and that would potentially not fit into the confines of a tabular layout, they also add complexity and increase the cognitive load for the user [36]. Tabular layouts, in contrast, make the association of all attributes to their item easy, but make it harder to see correlations between attributes or trends across the whole dataset.

As the *Keshif* example shows, tabular visualization techniques, such as *Taggle*, are an ideal complement to MCV systems: while selected attributes can be shown in dedicated views, for example, on a map or in a node-link layout, other attributes can be shown as part of the tabular visualization.

4.3 Aggregation Methods

Orthogonal to the design space discussed above are aggregation methods for the items within a table: representing the underlying distribution or statistical measures of a set of items is an important approach to increasing the scalability of visualization techniques. Aggregation can be applied to a whole dataset or to multiple groups of items and/or attributes separately. Examples of overview techniques are hierarchical parallel coordinates [37], which visualize cluster centroids rather than individual items, and *VisBricks* [7], which can visualize clusters using various techniques, including aggregation methods. An example that predominantly uses aggregations is *Keshif* [35], where a table of items is supplemented with multiple views showing distributions for interaction-driven exploration. To our knowledge, there is currently no interactive general tabular visualization technique that allows aggregation.

When working with large tabular data, not all data can be shown in detail, as the number of rows quickly exceeds the available display space. There are two potential remedies: scrolling and aggregation. While scrolling is common when working with

tables, it does not preserve the context of off-screen data items. Aggregation, in contrast, can be leveraged to preserve both details about a set of items in focus and context about the rest.

Elmqvist and Fekete [38] proposed several design guidelines for aggregation, including: *Visual Summary*—aggregates should convey information about the underlying data; *Discriminability*—aggregates can easily be distinguished from individual data items; and *Fidelity*—measures are taken to counteract artifacts of the aggregation process that misrepresent true effects. The aggregation techniques in *Taggle* were designed with these guidelines in mind.

There are various specialized tabular visualization tools that use aggregation in tabular layouts. *iHAT* [39] aggregates amino acid sequences and associated metadata using the most frequent category or the average to represent aggregated items, depending on the data type. Holzhüter et al. [40] use the average for numerical values for aggregates. Both techniques employ transparency to communicate fidelity (the higher the variation in a cell, the higher the transparency), but neither addresses fidelity well. The *Breakdown Visualization* technique by Conklin and North [41] aggregates rows or columns of a table based on a pre-existing aggregation hierarchy. Users can traverse the hierarchy and pivot through intersecting hierarchies.

Pivot tables are another way of dealing with the analysis of large multidimensional data. They enable rotation, that is, pivoting of various data dimensions, and with use of data aggregation, such as sum, average, or count, they provide a quick data overview. Pivot tables are widely available in spreadsheet tools such as Excel and Google Sheets, and similar operations are available as part of libraries and programming languages used to create static tabular figures. Tableau [16] employs pivot table principles for constructing nested matrices, where each matrix can then be represented visually in different ways. While pivot tables are well suited to, for instance, summing up the sales figures of multiple departments, they do not commonly support more nuanced aggregations, such as histograms and box plots.

Visual aggregation is often employed not only to minimize screen space, but also to provide additional contextual information. For example, tabular techniques such as *LineUp* [5] and *DataComb* [18] show histograms of column data atop each column, while *WeightLifter* [42] shows histograms of items that were filtered out at the bottom of the table.

5 TAGGLE CONCEPT

Taggle is a novel visualization technique designed for exploration and presentation of large and heterogeneous tables that are a combination of categorical, numerical, and textual vectors as well as homogeneous matrices. The ability to flexibly aggregate data subsets in both row and column directions is a core capability of a scalable table visualization technique. We start this section by introducing the hierarchical grouping and aggregation mechanism of *Taggle*, which provides the conceptual foundation for all further operations, such as filtering, sorting, and flexible encoding of the different subsets. On this basis, we then discuss the layout strategy that enables focus+context also for large tabular datasets.

5.1 Hierarchical Grouping and Aggregation

Taggle enables users to group items by using hierarchical combinations of attributes. The result of these nested grouping levels is an ordered tree where all leaves are items. Groups can be defined based on categorical attributes, numerical thresholds or

user selection. These groups can then be aggregated into compact summary representations, referred to as aggregated groups.

Non-leaf nodes at any level can be aggregated, which corresponds to a cut through the hierarchy as shown in Figure 2 (f). Each row of the resulting table then corresponds to either one item or one group. By adjusting the level at which to aggregate, users can dynamically control the level of detail of the rows when rendering the table [38].

Sorting (R2), filtering (R3), grouping (R4), and aggregating (R5) are topological operations that result in changes in the tree. In contrast, encoding (R1) and multiform (R6) correspond to property changes in the nodes. Figure 2 illustrates how the different topological operations alter the tree. By default, all rows of the table are added as nodes attached to a common root node. Filtering items from the table removes the corresponding nodes from the tree. Sorting of items or groups changes the order of nodes in the tree. Grouping items adds a level to the hierarchy.

To determine which rows to render, we use a level traversal strategy [38]. In an un-aggregated tree, this traversal strategy results in a list of rows that correspond to the items of the table. When a node is aggregated, the traversal stops at this point, and adds only one row for this non-leaf node, as indicated by the stippled horizontal line shown in Figure 2 (f).

Note that defining hierarchies and groups on the columns of a matrix (R9) works in the same way as for rows.

5.2 Layout Strategy

The last two requirements (R11, R12) introduced in Section 3 are conflicting. Therefore, our goal is to let the user control the layout in order to optimize it for the high-level tasks of (1) obtaining an **overview** and (2) seeing **details** for a subset of the items. In Taggle the user can toggle between the two layout modes.

Overview Mode

In overview mode, the goal is to show as many rows as possible in order to give users a good sense of the overall patterns and distributions (addressing requirement R11).

To achieve this, Taggle decreases the height of items until the whole table fits on the screen. Aggregated groups are shown using a fixed height. For items we define a minimum height of a single pixel, as lower values would introduce uncertainty due to interpolation artifacts [40]. Consequently, for tables that have more rows than can fit within the window, the table representation exceeds the available screen space and scroll bars are introduced. If this happens, the user has two options to make the table fit on the screen again: aggregating groups or filtering items.

When viewing the table in overview mode, users can still increase the level of detail for one or multiple items by selecting them. This is useful in cases where users spot items of interest that they want to inspect in detail.

Detail Mode

The goal of the detail mode is to allow users to see all details for the items that fit on the screen (R12), including labels, numerical values, and category names. While this maximizes the readability of items (see requirement R12), it comes at the cost of reducing the number of visible items (R11). In detail mode, all items have a uniform height. This, however, causes the table to rapidly outgrow the window, making it necessary to scroll to access the non-visible part of the table or to reduce the height of the table by aggregating groups.

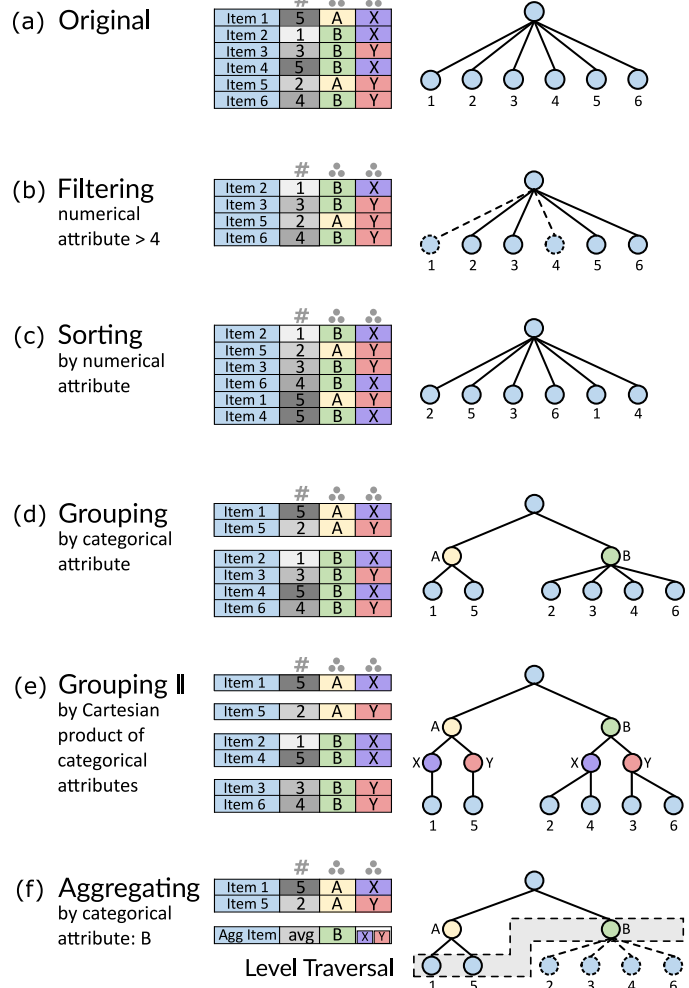


Fig. 2: Illustration of topological operations on a heterogeneous table (a) consisting of numerical (#) and categorical (♣) attributes and their results reflected in the aggregation hierarchy: sorting (b), filtering (c), grouping by a single categorical attribute (d), grouping by the Cartesian product of two categorical attributes (e), and aggregating (f).

6 VISUALIZATION AND INTERACTION DESIGN

The Taggle interface consists of two parts, as shown in Figure 1: the main **table view** (a) for visualizing the data and a **data selection panel** (b) for selecting the vectors and matrices to be shown in table view and for filtering. Below, we introduce the visual elements and interactions in detail, together with justifications of our design decisions.

The data selection panel provides access to all loaded numerical, categorical, text, and matrix attributes. When a vector or a matrix has been selected from a list of all loaded attributes, it is added as the last column to the table view. The user can change the order of columns via drag and drop. For each column that is present in the table view, the data selection panel shows a visual summary of the data in the form of a histogram for suitable datasets.

The column headers provide means for sorting, changing visual encoding and, where applicable, grouping. Users can select one or multiple items that will be highlighted by brushing over them while holding down the mouse or by checking a selection box offered for each item. In the overview, selected items will be increased to a predefined fixed height that allows labels to be shown.

6.1 Sorting

Sorting is a simple way of identifying minima and maxima in columns and relationships between columns. Offering flexible means for determining the sorting of rows in a table visualization is one of the most essential requirements (see R2). In addition to providing the standard feature of sorting items in ascending or descending order by a single numerical, textual, or categorical vector, Taggle enables users to sort items hierarchically, where a top-level column determines the initial sorting, a second column breaks ties from the initial sorting, and so on. This is particularly useful when sorting by categorical columns. Users can also sort matrix columns by specifying a statistical measure (minimum, maximum, lower and upper quartile, median, mean) as sorting criterion.

While other table visualizations such as the Visual Spreadsheet [24] sort attributes hierarchically based on the order of the columns, we decided to separate the sorting from the layout. If no grouping is active, we use the order of attributes in the data selection panel to define the hierarchical sorting. First, the rows are sorted by the attribute shown at the top of the panel. If multiple items have the same value or are in the same category, the second attribute will be taken to resolve the tie. This strategy is applied recursively, as illustrated in Figure 1, where the countries are sorted first by two categorical attributes (*continent* and *human development index*) and then by a numerical vector (number of *people knowing they have HIV*).

Which columns are used for sorting is shown by highlighting the sort button and by a number identifying the rank of the attribute in the data selection panel.

6.2 Filtering and Transforming Data

In Taggle, filters can be defined by interacting with the histograms in the data selection panel either by brushing a range in the case of numerical data (Figure 1 (b), *people knowing they have HIV*) or by selecting categories that are to be removed from the table (Figure 1 (b), *continent*). Textual data can be filtered by string matching or by a regular expression. In addition, users can filter out items with missing values.

Flexibly transforming how raw data is mapped to a visual element (see R8) is important, for instance, to refine scale after filtering. For this purpose, Taggle uses the visual mapping editor introduced in LineUp [5], which offers a set of predefined data mapping functions such as inverse and logarithmic mapping, and a user interface for defining scales and mapping functions.

6.3 Grouping

Being able to stratify tables into meaningful groups is not only an important feature for structuring tabular data (see requirement R4), but also an essential prerequisite for aggregation operations in Taggle.

As grouped elements appear in adjacent rows, grouping is related to sorting, but Taggle separates these operations in order to enable more fine-grained control of groups. We leverage categorical vectors to group datasets, which allows us, for example, to group countries by *continent*. Numerical vectors can be used to split the dataset into groups based on bins. Users can also split datasets into groups by selecting items. Combining multiple hierarchically sorted grouping attributes creates fine-grained groups that correspond to the Cartesian product of the constituting categories. In practice, we

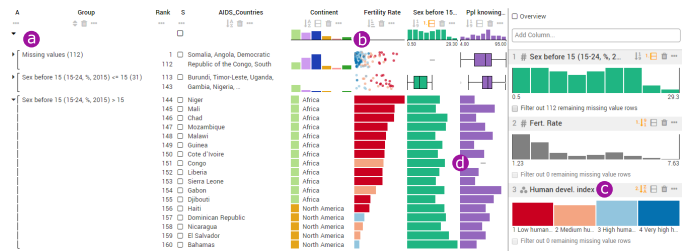


Fig. 3: Taggle table showing countries grouped by bins of the percentage of the population who had *sex before the age of 15* (a). Countries with over 15% are shown in detail mode. The *fertility rate* values (b) are colored according to the *human development index* (c), showing the correlation between the two attributes. Missing values are encoded using dash (d).

found that two to three grouping levels are sufficient, as more lead to fragmented groups.

A dedicated group column summarizes how the group is defined and how many items are contained. In Figure 1, for instance, the combination of the attributes *continent* and *the human development index* constitute the grouping, which is indicated in the first column. This column can be utilized to sort the groups by their name or by their size (i.e., the number of items they contain).

As grouping establishes a visual order in the table, attributes that define a grouping are moved to the top of the data selection panel and to the corresponding place in the sorting hierarchy. By default, the order of attributes in the sorting hierarchy corresponds to the grouping hierarchy and the visual order of attributes in the table. An exception to this is the case in which items are grouped based on a binned numerical attribute, but sorted according to a different attribute. Figure 3 illustrates a case in which the countries were first grouped based on percentage of women who had *sex before the age of 15* with a threshold set to 15 percent (Figure 3 (a)), but sorted according to *fertility rate* (Figure 3 (b)). Interestingly, only African and North American countries fell within the group with high percentages of *sex before 15*. Sorting the table by *fertility rates* shows a clear difference between the countries of the two continents, with North American countries having much lower fertility rates than the African countries in this group. This correlates to the level of *human development index*.

6.4 Matrices

Adding a matrix to a table visualization introduces a second key for the columns of the matrix. To support the requirement for bidirectional matrix operations (R9), we allow grouping of matrix columns based on this key. The individual groups of columns are then treated as separate matrices—they can be manually reordered, aggregated, and sorted, and the visual encoding of each group can be adjusted individually. For example, the years in the *new HIV infections per 1,000 people* matrix and the *AIDS-related deaths per 1,000 people* matrix in Figure 1 (c) introduce *years* as the second key, which is then used to group these matrices by *decades*. Here, the 2010s use a different visual encoding for the groups.

6.5 Aggregation

Groups can be leveraged to aggregate the data: instead of representing each row in the dataset by a row in the table, we aggregate the grouped rows into a single row summarizing all items contained. Aggregated groups are assigned a uniform height that is about twice

that of a row shown in detail mode. Aggregating items into groups commonly mandates changing the visual encoding. For example, instead of bar plots for individual items, we show a histogram that represents the whole group. The visual encoding of grouped cells can also be altered to, for example, replace a histogram with a box plot (see Figure 1d). The user can toggle the aggregation for each group of rows across all columns.

6.6 Encoding and Multiform Visualizations

The table view encodes each selected vector or matrix using one of multiple alternative visual encodings suitable for the data type. This includes bars, dots, proportional symbols, or brightness for numerical data, color or positional/matrix encoding for categorical data, and heatmaps for matrices (addressing requirement R1).

The visual encoding for each column can be changed on demand [7], supporting the multiform principle (R6). For example, the default bar encoding a single numerical attribute can be interactively changed to a proportional symbol, if desired. Dedicated visual encodings are used for aggregates: box plots and histograms show the distribution of numerical values; stacked bars and histograms show relative frequencies of categories in an aggregate. A list of text items is represented as a truncated list of examples. Figure 4 gives an overview of the visual encodings available for numerical, categorical, and textual attributes with and without aggregation. Figure 5 summarizes how a matrix can be aggregated in column and row directions.

While our implementation is not comprehensive, we deliberately limit the range of visual encodings to choices that either offer superior perceptual properties or are very compact, thus allowing users to choose between the perceptual accuracy and space utilization. We deliberately do not offer visual encodings that we consider to be problematic. For example, a bar representing an average of a group does not communicate any variability and is therefore not a suitable visualization for an aggregated vector [43].

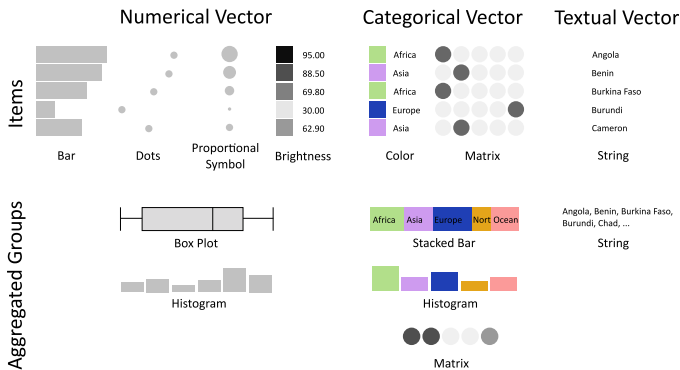


Fig. 4: Vector visualization techniques for items and aggregated groups by data type. Numerical items can be encoded with bars, dot plots, proportional symbols, or brightness. For categorical vectors, we offer color encoding and a matrix representation, where each category is shown as present/absent in a binary column. All items can also be displayed as strings. Numerical vectors can be aggregated into box plots and histograms. Distributions of categorical values can be shown as a histogram, a stacked bar, or an aggregated matrix with brightness encoding frequency of individual categories in the group. An aggregated textual vector shows examples of the group members.

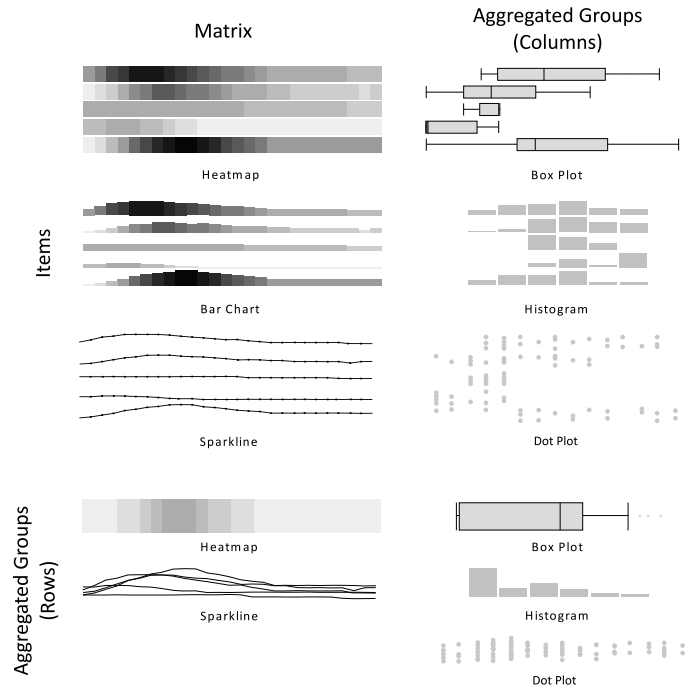


Fig. 5: Matrix visualization techniques and aggregation principles. Matrix items can be encoded using brightness / as a heatmap, as bar charts, and as sparklines. Matrices can be aggregated in both column and row directions. When a matrix is aggregated in the column direction, a group of matrix columns within one row is merged into a single cell. The aggregated values can then be visualized using box plots, histograms and dot plots. When a matrix is aggregated in row direction, a group of rows is merged into one row. Values of aggregated rows can be displayed using a heatmap and superimposed sparklines. A matrix aggregated in both directions is encoded using a box plot, histogram, or dot plot of all matrix values.

Compact Encodings

When rows are down-scaled to fit into the available vertical screen space, we take various measures to adapt the visualization to the diminished space, as illustrated in Figure 6. We not only make the visual representations smaller, but also reduce details and/or adapt the visualization. In the compact representation of box plots, for instance, we fill the available vertical space at the position of the box and indicate the start and end of the whiskers by drawing vertical tick marks. However, some visualizations, such as strings and proportional symbols, do not have an adequate down-scaled complement, in which case they are not rendered.

We chose a dash to encode missing values (Figure 3 (d)). We also considered a dedicated color, but dashes have the advantage that their visual saliency is lower (i.e., they do not draw as much attention), and yet they are clearly visible at all levels of detail.

6.7 Combining Columns

Giving the users the possibility to flexibly combine columns (R7) supports a wide range of tasks. The user can interactively create combined columns either by dragging existing ones on an empty container or by dragging one column onto another. The possible combinations are specific to the data type of the column.

The most basic combined column is a **nested column**, shown in Figure 7 (a) and (e). It encloses multiple individual columns by

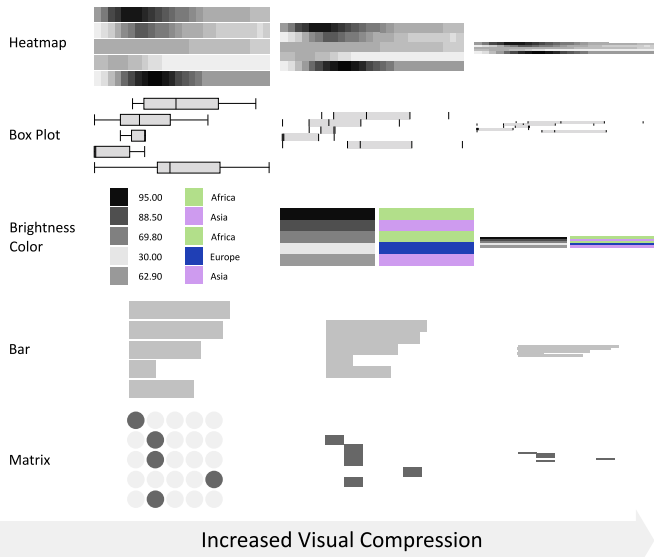


Fig. 6: Example of encodings at different scales. In the first column, the items are displayed at full height with white space separating the rows. If a textual label is part of the visualization, it is displayed at a readable size. Compact representations (columns two and three) remove white space and string labels. Some visualizations, such as the box plot or the categorical matrix, are simplified to account for the limited space.

adding a joint header above all columns contained. Nested multiple columns are useful for creating semantic groups. It is the most flexible column combiner that works for all types and can mix columns of different types.

Taggle also enables users to create **stacked columns** [5], [25] by combining two or multiple numerical columns to create a weighted sum of the items and where the individual contributions are represented as *stacked bars* (see Figure 7 (b)). Users can interactively change the weights of individual columns by adapting their widths. Stacked columns can be used to create a “score”, which, in turn, can be used to create rankings. Aggregate representations for stacked columns are shown as box plots, where the values feeding the box plots are the weighted sums of the composing values.

To enable a more effective comparison of items across multiple columns, an **interleaved column** (Figure 7 (c)) stacks the encoded values from multiple numerical columns vertically. Depending on whether the row is an item or group, the stacked representations can be made up from bars or dots, or, in case an aggregate is interleaved, of a box plot.

With **imposition columns** users can color the visual marks (bar, proportional symbol, etc.) of a numerical vector by the color coding of a categorical vector, as shown in Figure 7 (f).

Taggle also enables more complex combinations, based on a set of predefined functions, such as minimum, maximum (Figure 7 (d)), and mean, for combining multiple numerical vectors into a single numerical column. In addition, users can add **scripted columns** that allow them to define their own functions via a scripting interface [5].

6.8 Animated Transitions

We support users in understanding changes in the visualization by applying animated transitions [9], as demonstrated in the accompanying video. The prototype implementation incorporates

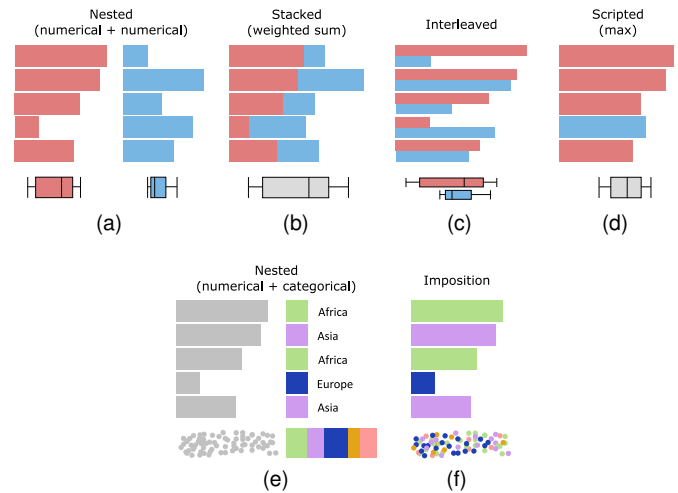


Fig. 7: Possible column combinations: (a, e) nested column that semantically groups columns of various types, (b) stacked column that creates a stacked bar plot based on multiple weighted numerical columns, (c) interleaved column that stacks the visualizations of multiple numerical columns, (d) scripted column that, in this case, visualizes only the maximum values of selected columns, and (f) column imposition where the marks of a numerical column are colored by the imposed categorical column.

smooth transitions for the switch between overview and detail mode as well as for changes resulting from filter, sort, and aggregation operations.

Instead of simply morphing item position, we apply staged transitions, where animations are split into multiple phases [9]. In the first phase of a filter animation, for instance, we fade out the filtered rows and then move up the remaining rows of the table to fill the white space. This is designed to help users understand why rows outside the viewport become visible in the bottom of the table. Similarly, when users aggregate a group, we first fade out the aggregated rows, then gradually reduce the height of the empty area to the fixed height of the aggregated group, and finally fill the area by fading in the aggregated visualizations across all columns.

6.9 Sorting and Grouping of Column Subsets

While Taggle focuses primarily on tabular visualization, keeping items in constant rows across all columns, it also supports splitting a table into multiple segments and sorting and grouping each instance independently. To encode the relationships between table segments, we utilize slope graphs for connecting individual items of the tables compared [44, p. 156] and bands for showing relationships between aggregated groups [11], [12], de-facto enabling users to create hybrid tabular/overview representations, and in the extreme, even visualization techniques such as parallel sets [45].

7 IMPLEMENTATION

The Taggle feature set is fully integrated into the *LineUp.js* library, which is written in TypeScript and available as open source via Github⁴. A demo version can be accessed at <https://taggle.caleydoapp.org/>.

4. <https://github.com/Caleydo/lineupjs/>

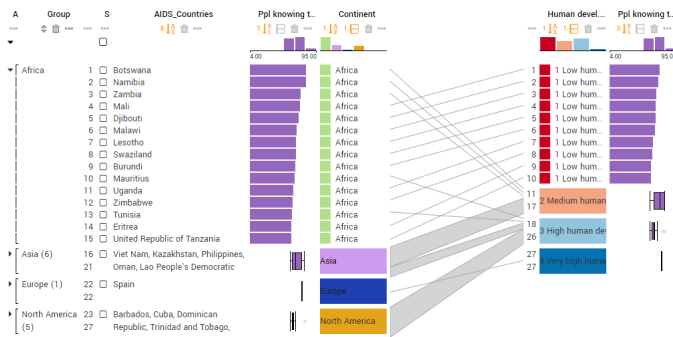


Fig. 8: Comparison of two table segments. The segment on the left shows countries grouped by *continent*. The segment on the right visualizes countries grouped by *human development index*. Both table segments are ranked by the number of *people knowing they have HIV*. The steeper the angle of the lines connecting the two instances, the greater is the change in the ranking. Bands show relationships between aggregated groups.

Making Taggle available as an open-source library increases potential adoption of the technique. The library can also be embedded in Jupyter Notebook⁵ and used as an HTML widget for R⁶, which allows integration into Shiny applications⁷ and R Notebooks⁸.

In the demo application, users can switch between multiple preloaded datasets and upload new datasets in CSV format. The Taggle table can be saved and restored via the Github Gist service or downloaded as a CSV file.

8 CASE STUDY: DRUG TARGET DISCOVERY

We evaluated Taggle by means of a case study, a method widely used to validate visualization systems [46], [47]. The case study summarizes an analysis session carried out by a collaborator working in a drug discovery team in a pharmaceutical company. For the case study, we integrated Taggle with the Ordino Target Discovery Platform that provides access to the required cancer genomics data⁹.

In order to identify potential drug targets in a set of tumor types, the analyst performs experiments with cancer cell lines—cultured cells that are derived from tumors and that can proliferate indefinitely in the laboratory. These cell lines are characterized by various properties, such as tumor type (lung cancer, prostate cancer, etc.) and the set of genes that are mutated.

One very important gene in the context of cancer is *TP53*. It encodes the p53 protein, whose presence is known to suppress the uncontrolled division of cells. However, when *TP53* is mutated—which is the case for over 50% of cancer patients—it can lose its suppressing function, which results in tumor growth. Due to its important role, scientists want to know whether *TP53* is mutated in a set of cell lines. However, the mutation status of *TP53* is not always known. It has recently been shown that the mean expression level (expression is a measure of the activity of genes) of 13 genes that are biologically related to *TP53* is correlated with its mutation

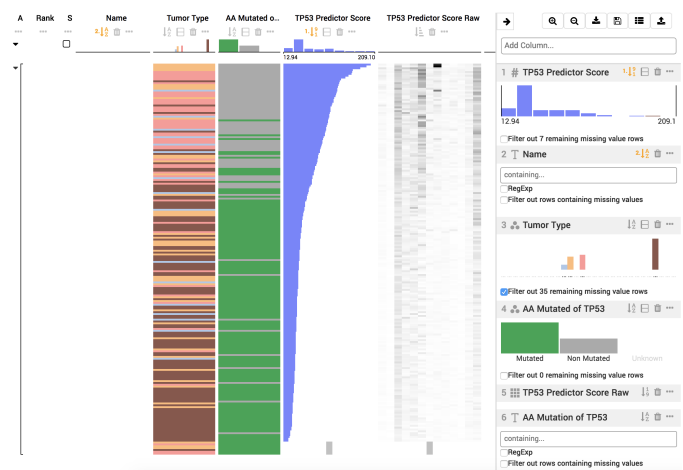


Fig. 9: After sorting the cell lines by the *TP53 predictor score* (blue), the analyst notices that cell lines with a low average score are much more likely to be mutated (green). From this the analyst concludes that predicting the mutation status based on the average expression of the 13 genes that constitute the predictor score works reasonably well.

status. The expression level of these genes can hence be used to predict the mutation status of *TP53* [48].

In this case study, the analyst first wants to find out how well this predictor works for the set of cell lines contained in the database. Based on this knowledge and other criteria, the analyst then wants to select cell lines for a wet-lab experiment.

The analyst starts by loading a list of 1,009 cell lines from the public CCLE dataset [49] into Taggle. By default, the table contains a textual column representing cell lines' *name* and a categorical column indicating *tumor type*. Since only a subset of tumor types is of interest, the analyst filters for *astrocytoma/glioblastoma*, *bone sarcoma*, *melanoma*, and *non-small-cell lung cancer (NSCLC)*, after which 255 cell lines remain.

As the analyst wants to investigate the *TP53* gene, he loads a categorical column with the *mutation status* (mutated vs. non-mutated) and a textual column that provides further details about the mutation (if present). According to the mutation histogram in the data selection panel, the status is unknown for 59 cell lines. To investigate the effectiveness of the 13 genes in predicting the *TP53* status, the analyst loads the average expression of these genes together with a matrix column containing the individual expression values. Furthermore, he hides cell lines with unknown mutation status. After sorting the table by average expression in descending order and switching to the overview (see Figure 9), the analyst observes the overall good correlation between expression and mutation status: there is a clear enrichment of *TP53* mutants among the cell lines with low score.

In order to test whether the correlation is present for all selected tumor types, the analyst groups the table by tumor type. He observes that the prediction seems to work particularly well for the *astrocytoma/glioblastoma* cell lines (almost perfect separation between mutated and non-mutated) and further investigates this by also stratifying by mutation status and collapsing all groups (see Figure 10). The expression box plots show good separation for *astrocytoma/glioblastoma* and *melanoma*, but the expression ranges are overlapping for *NSCLC*.

Having confirmed that the prediction of the *TP53* mutation status works reasonably well in several tumor types, the analyst

5. <https://jupyter.org/>

6. <https://www.htmlwidgets.org/>

7. <http://shiny.rstudio.com/>

8. http://rmarkdown.rstudio.com/r_notebooks.html

9. <https://ordino-taggle.caleydoapp.org/>

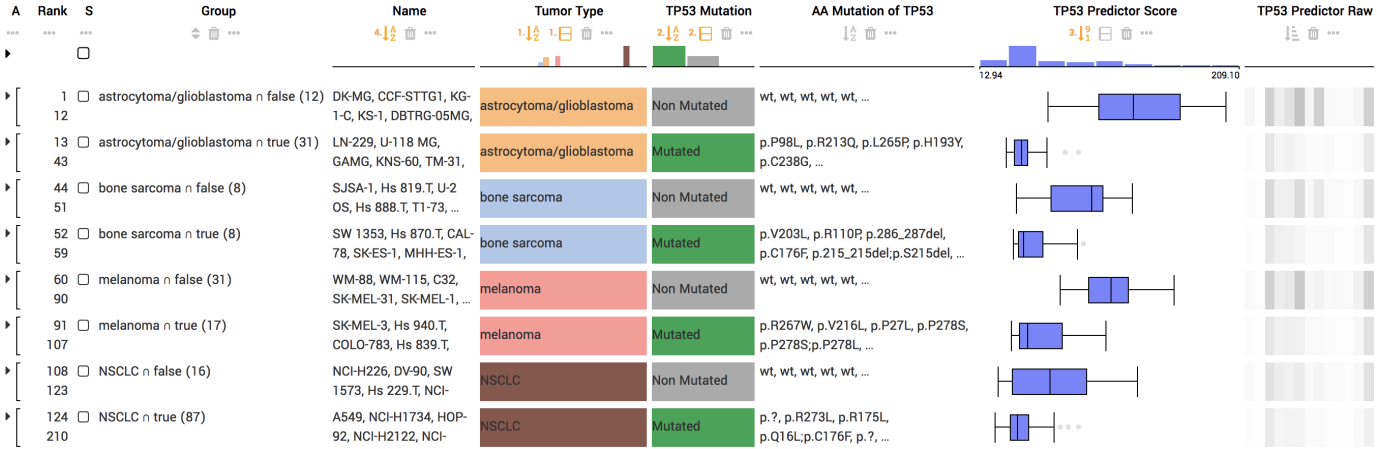


Fig. 10: The analyst groups the cell lines first by the attribute *tumor type* and then by *TP53 mutation status*. For the tumor type *astrocytoma/glioblastoma* the box plots representing the *TP53 predictor score* show a clear separation between the groups *mutated* and *non-mutated*. For the other tumor types the whiskers of the box plots overlap, indicating that the predictor score does not work as effectively.

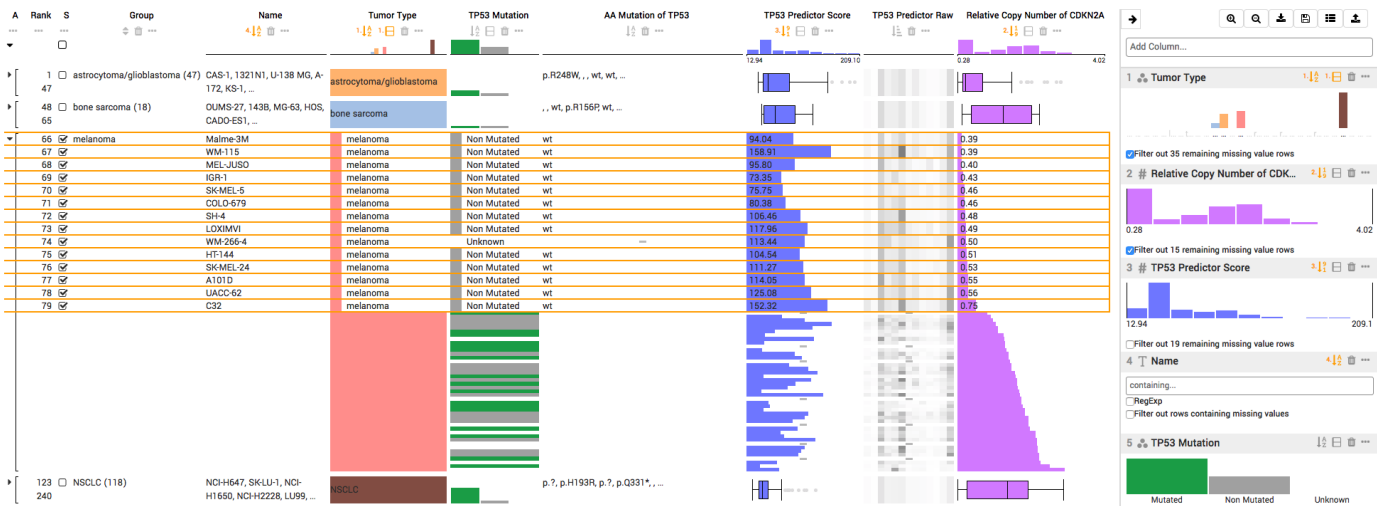


Fig. 11: Continuing from the visualization state shown in Figure 10, the analyst removes the grouping on the *TP53 mutation* column and unaggregates the *melanoma* group to inspect the cell lines in further detail. With the goal to find cell lines for a wet-lab experiments, the analyst adds the *copy number value* of *CDKN2A* as an additional column (shown in purple). Finally, he selects cell lines that have a low *copy number value* and are either non-mutated or have unknown mutation status and a *TP53 predictor score* above 110.

wants to select a set of cell lines for a wet-lab experiment. He is interested in *melanoma* cell lines that have no *TP53* mutation. Furthermore, the activity of *CDKN2A*, another important tumor suppressor gene, should be impaired due to a reduced number of *CDKN2A* gene copies in the genome. The analyst removes the mutation status grouping, includes cell lines for which it is unclear whether *TP53* is mutated, and unfolds the *melanoma* cell lines group. Based on the ranking, he decides to consider all cell lines with unknown *TP53* mutation status and a *TP53* predictor score greater than 110 as non-mutated. He adds a column with the *CDKN2A* relative copy number, sorts by it in ascending order, and filters out missing data. Finally, he selects the top hits of the resulting list (see Figure 11). All of these cell lines fulfill the analyst's requirements.

9 DISCUSSION AND LIMITATIONS

Revisiting our discussion of visualization techniques for tabular data (overview, projection, and tabular techniques), we argue that

Taggle is primarily a tabular visualization technique, as it retains a tabular layout and encodes data within a cell, but also has some aspects of an overview technique due to its capabilities to aggregate and its ability to sort and group subsets of columns independently. Interactive definition of groups and their aggregation in summary visualizations, such as box plots and histograms, provides a meaningful overview even for large data sets and enables an intuitive comparison of grouped data subsets (Figure 10). At the same time, Taggle enables the exploration of items at a detailed level to identify their precise properties (Figure 11).

This combination sets Taggle apart from existing tabular techniques, which provide only a coarse overview of the items (e.g., using the lens technique, which is insufficient for representation or comparison of large data sets) or lack interactivity, which is essential to the exploration process.

Scalability

In detail mode, Taggle scales to more than 100,000 rows. We achieve this performance by leveraging rendering optimizations,

which ensure that only visible rows are rendered. However, in overview mode, which tries to fit all items on the screen, this performance optimization is no longer effective. Hence, for tables with more than 1,000 rows and matrices with many columns, the rendering performance decreases due to the large number of DOM elements.

The problem can be alleviated by aggregating all groups in the table by default. However, this approach is in opposition to the idea of a space-filling overview. This technical limitation could be mitigated by replacing the DOM-based rendering with an HTML5 Canvas implementation.

Stacking of Matrices and Vectors

Our current prototype supports grouping of matrix columns based on a categorical attribute (see Figure 1 (c)), but provides no means of sorting and filtering the matrix columns. Furthermore, it is not possible to stack additional attributes on top of a matrix, as done, for instance, in Figures 4 and 6 presented in [50]. Due to these technical limitations, which we plan to address in future versions, we consider R9 in Table 1 only as partly addressed.

Automatic Aggregation

In the design process, we investigated methods for automatically aggregating rows and columns, with the goal of increasing scalability. For instance, to make space for brushed rows that are shown with increased height, we tried to automatically aggregate groups that were not affected by the brush. We found, however, that users had difficulties in understanding such changes in the visual representation. This may be due to perceptual phenomena, such as change blindness, or due to the fact that it can be hard to understand changes that the user did not actively trigger. As part of future work, we plan to implement and evaluate a recommendation approach that suggests possible layout changes without automatically applying them.

10 CONCLUSION AND FUTURE WORK

We have presented Taggle, a scalable tabular data visualization technique for combining, filtering, and aggregating attributes of a heterogeneous dataset. Taggle is unique among tabular data visualization techniques due to its ability to dynamically aggregate subsets of a table and due to its flexibility with respect to data types and visual encoding. We argue that Taggle is useful for both interactive exploration and presentation purposes.

Taggle is broadly applicable, as tabular/spreadsheet datasets are widely used, and our implementation goes beyond a research prototype, providing data import/export, creating persistent states, etc.

Much of Taggle's functionality could also be integrated with traditional spreadsheet applications, combining the powerful logic and math enabled by spreadsheets, and the ease of exploration and discovering trends and outliers in Taggle.

As part of future work we plan to integrate data-driven guidance capabilities into Taggle, as implemented in StratomeX [51]. Following the idea of guided visual exploration, we plan to assist users in finding correlated attributes or similar groups based on their input.

ACKNOWLEDGMENTS

We thank Bikram Kawan for his contributions to the initial prototype implementation as well as Christian Haslinger and Andreas Wernitznig for providing valuable conceptual feedback. This work was supported in part by Boehringer Ingelheim Regional Center Vienna, the State of Upper Austria (FFG #851460), and the US National Institutes of Health (U01 CA198935).

REFERENCES

- [1] J. H. Claessen and J. J. van Wijk, "Flexible Linked Axes for Multivariate Data Visualization," *IEEE Transactions on Visualization and Computer Graphics (InfoVis '11)*, vol. 17, no. 12, pp. 2310–2316, 2011.
- [2] S. Liu, D. Maljovec, B. Wang, P. T. Bremer, and V. Pascucci, "Visualizing High-Dimensional Data: Advances in the Past Decade," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 3, pp. 1249–1268, Mar. 2017.
- [3] R. Rao and S. K. Card, "The table lens: merging graphical and symbolic representations in an interactive focus + context visualization for tabular information," in *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI '94)*. ACM, 1994, pp. 318–322.
- [4] C. Perin, P. Dragicevic, and J. D. Fekete, "Revisiting Bertin Matrices: New Interactions for Crafting Tabular Visualizations," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 2082–2091, 2014.
- [5] 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 (InfoVis '13)*, vol. 19, no. 12, pp. 2277–2286, 2013.
- [6] Z. Gu, R. Eils, and M. Schlesner, "Complex heatmaps reveal patterns and correlations in multidimensional genomic data," *Bioinformatics*, vol. 32, no. 18, pp. 2847–2849, 2016. [Online]. Available: <https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btw313>
- [7] A. Lex, H.-J. Schulz, M. Streit, C. Partl, and D. Schmalstieg, "VisBricks: Multifom Visualization of Large, Inhomogeneous Data," *IEEE Transactions on Visualization and Computer Graphics (InfoVis '11)*, vol. 17, no. 12, pp. 2291–2300, 2011.
- [8] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein, "Cluster analysis and display of genome-wide expression patterns," *Proceedings of the National Academy of Sciences USA*, vol. 95, no. 25, pp. 14 863–14 868, 1998.
- [9] J. Heer and G. G. Robertson, "Animated Transitions in Statistical Data Graphics," *IEEE Transactions on Visualization and Computer Graphics (InfoVis '07)*, vol. 13, no. 6, pp. 1240–1247, 2007.
- [10] N. Henry, J. D. Fekete, and M. J. McGuffin, "NodeTriX: a Hybrid Visualization of Social Networks," *IEEE Transactions on Visualization and Computer Graphics (InfoVis '07)*, vol. 13, no. 6, pp. 1302–1309, 2007.
- [11] A. Lex, M. Streit, H.-J. Schulz, C. Partl, D. Schmalstieg, P. J. Park, and N. Gehlenborg, "StratomeX: Visual Analysis of Large-Scale Heterogeneous Genomics Data for Cancer Subtype Characterization," *Computer Graphics Forum (EuroVis '12)*, vol. 31, no. 3, pp. 1175–1184, 2012.
- [12] S. Gratzl, N. Gehlenborg, A. Lex, H. Pfister, and M. Streit, "Domino: Extracting, Comparing, and Manipulating Subsets across Multiple Tabular Datasets," *IEEE Transactions on Visualization and Computer Graphics (InfoVis '14)*, vol. 20, no. 12, pp. 2023–2032, 2014.
- [13] M. Streit, S. Gratzl, M. Gillhofer, A. Mayr, A. Mitterecker, and S. Hochreiter, "Furby: Fuzzy Force-Directed Bicluster Visualization," *BMC Bioinformatics*, vol. 15, no. Suppl 6, p. S4, 2014. [Online]. Available: <http://www.biomedcentral.com/qc/1471-2105/15/S6/S4>
- [14] J. Stahnke, M. Dörk, B. Müller, and A. Thom, "Probing Projections: Interaction Techniques for Interpreting Arrangements and Errors of Dimensionality Reductions," *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 1, pp. 629–638, 2016.
- [15] "Excel 2016." [Online]. Available: <https://products.office.com/en-us/excel/>
- [16] C. Stolte, D. Tang, and P. Hanrahan, "Polaris: A System for Query, Analysis, and Visualization of Multidimensional Databases," *Communications of the ACM*, vol. 51, no. 11, pp. 75–84, 2008. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1400234>
- [17] M. Spenke and C. Beilken, "InfoZoom - Analysing Formula One racing results with an interactive data mining and visualisation tool," in *Proceedings of the Conference on Data Mining*, 2000, pp. 455–464.
- [18] C. Polis, "DataComb: An interface for combing through tabular datasets | Chris Polis, ByteMuse.com." [Online]. Available: <http://www.bytemuse.com/post/data-comb-visualization/>

- [19] “Google Sheets.” [Online]. Available: <https://www.google.com/sheets/about/>
- [20] “Apache OpenOffice Calc.” [Online]. Available: <https://www.openoffice.org/product/calc.html>
- [21] M. Spenke, C. Beilken, and T. Berlage, “FOCUS: The Interactive Table for Product Comparison and Selection,” in *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST '96)*. ACM, 1996, pp. 41–50. [Online]. Available: <http://doi.acm.org/10.1145/237091.237097>
- [22] “InfoZoom.” [Online]. Available: <https://www.infozoom.com/en/>
- [23] M. Goldman, “Visual Spreadsheet,” Oct. 2017. [Online]. Available: <http://xena.ucsc.edu/visual-spreadsheet/>
- [24] C. Tyner, G. P. Barber, J. Casper, H. Clawson, M. Diekhans, C. Eisenhart, C. M. Fischer, D. Gibson, J. N. Gonzalez, L. Guruvadoo, M. Haussler, S. Heitner, A. S. Hinrichs, D. Karolchik, B. T. Lee, C. M. Lee, P. Nejad, B. J. Raney, K. R. Rosenbloom, M. L. Speir, C. Villarreal, J. Vivian, A. S. Zweig, D. Haussler, R. M. Kuhn, and W. J. Kent, “The UCSC Genome Browser database: 2017 update,” *Nucleic Acids Research*, vol. 45, no. D1, pp. D626–D634, 2017. [Online]. Available: <https://academic.oup.com/nar/article/45/D1/D626/2605793/The-UCSC-Genome-Browser-database-2017-update>
- [25] G. Carenini and J. Lloyd, “ValueCharts: analyzing linear models expressing preferences and evaluations,” in *Proceedings of the Conference on Advanced Visual Interfaces (AVI '04)*, ser. AVI '04. ACM, 2004, pp. 150–157. [Online]. Available: <http://doi.acm.org/10.1145/989863.989885>
- [26] A. Lex, M. Streit, C. Partl, K. Kashofer, and D. Schmalstieg, “Comparative Analysis of Multidimensional, Quantitative Data,” *IEEE Transactions on Visualization and Computer Graphics (InfoVis '10)*, vol. 16, no. 6, pp. 1027–1035, 2010.
- [27] M. Kern, A. Lex, N. Gehlenborg, and C. R. Johnson, “Interactive Visual Exploration And Refinement Of Cluster Assignments,” *bioRxiv*, p. 123844, 2017. [Online]. Available: <http://biorxiv.org/content/early/2017/04/04/123844>
- [28] J. Seo and B. Shneiderman, “Interactively Exploring Hierarchical Clustering Results,” *Computer*, vol. 35, no. 7, pp. 80–86, 2002.
- [29] H.-M. Wu, Y.-J. Tien, and C.-h. Chen, “GAP: A graphical environment for matrix visualization and cluster analysis,” *Computational Statistics & Data Analysis*, vol. 54, no. 3, pp. 767–778, 2010. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0167947308004349>
- [30] G. Caraux and S. Pinloche, “PermutMatrix: a graphical environment to arrange gene expression profiles in optimal linear order,” *Bioinformatics*, vol. 21, no. 7, pp. 1280–1281, 2005. [Online]. Available: <https://academic.oup.com/bioinformatics/article/21/7/1280/269055/PermutMatrix-a-graphical-environment-to-arrange>
- [31] N. F. Fernandez, G. W. Gundersen, A. Rahman, M. L. Grimes, K. Rikova, P. Hornbeck, and A. Ma’ayan, “Clustergrammer, a web-based heatmap visualization and analysis tool for high-dimensional biological data,” *Scientific Data*, vol. 4, p. sdata2017151, 2017. [Online]. Available: <https://www.nature.com/articles/sdata2017151>
- [32] H. Wickham, “ggplot2,” *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 3, no. 2, pp. 180–185, 2011. [Online]. Available: <http://onlinelibrary.wiley.com/doi/10.1002/wics.147/full>
- [33] J. D. Hunter, “Matplotlib: A 2d Graphics Environment,” *Computing in Science Engineering*, vol. 9, no. 3, pp. 90–95, May 2007.
- [34] C. Weaver, “Building Highly-Coordinated Visualizations in Improve,” in *Proceedings of the IEEE Symposium on Information Visualization (InfoVis '04)*. IEEE, 2004, pp. 159–166.
- [35] M. A. Yalcin, N. Elmquist, and B. B. Bederson, “Keshif: Rapid and Expressive Tabular Data Exploration for Novices,” *IEEE Transactions on Visualization and Computer Graphics*, vol. PP, no. 99, pp. 1–1, 2017.
- [36] M. Q. W. Baldonado, A. Woodruff, and A. Kuchinsky, “Guidelines for using multiple views in information visualization,” in *Proceedings of the ACM Conference on Advanced Visual Interfaces (AVI '00)*. ACM, 2000, pp. 110–119.
- [37] Y.-H. Fua, M. O. Ward, and E. A. Rundensteiner, “Hierarchical parallel coordinates for exploration of large datasets,” in *Proceedings of the IEEE Conference on Visualization (Vis '99)*. IEEE Computer Society Press, 1999, pp. 43–50.
- [38] N. Elmquist and J.-D. Fekete, “Hierarchical Aggregation for Information Visualization: Overview, Techniques, and Design Guidelines,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 3, pp. 439–454, 2010.
- [39] J. Heinrich, C. Vehlou, F. Battke, G. Jäger, D. Weiskopf, and K. Nieselt, “iHAT: interactive hierarchical aggregation table for genetic association data,” *BMC bioinformatics*, vol. 13, no. 8, p. 1, 2012. [Online]. Available: <http://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-13-S8-S2>
- [40] C. Holzhüter, A. Lex, D. Schmalstieg, H.-J. Schulz, H. Schumann, and M. Streit, “Visualizing Uncertainty in Biological Expression Data,” in *Proceedings of the SPIE Conference on Visualization and Data Analysis (VDA '12)*, vol. 8294. IS&T/SPIE, 2012, p. 829400.
- [41] N. Conklin, S. Prabhakar, and C. North, “Multiple Foci Drill-down Through Tuple and Attribute Aggregation Polyarchies in Tabular Data,” in *Proceedings of the IEEE Symposium on Information Visualization (InfoVis '02)*. IEEE, 2002, pp. 131–134. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1173158
- [42] S. Pajer, M. Streit, T. Torsney-Weir, F. Spechtenhauser, T. Möller, and H. Piringer, “WeightLifter: Visual Weight Space Exploration for Multi-Criteria Decision Making,” *IEEE Transactions on Visualization and Computer Graphics (InfoVis '16)*, vol. 23, no. 1, pp. 611–620, 2017.
- [43] M. Streit and N. Gehlenborg, “Points of View: Bar charts and box plots,” *Nature Methods*, vol. 11, no. 2, pp. 117–117, 2014. [Online]. Available: <http://www.nature.com/nmeth/journal/v11/n2/full/nmeth.2807.html>
- [44] E. Tufte, *The Visual Display of Quantitative Information*, 2nd ed. Cheshire, CT, USA: Graphics Press, 1983.
- [45] R. Kosara, F. Bendix, and H. Hauser, “Parallel Sets: Interactive Exploration and Visual Analysis of Categorical Data,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 4, pp. 558–568, 2006.
- [46] M. Sedlmair, M. Meyer, and T. Munzner, “Design Study Methodology: Reflections from the Trenches and the Stacks,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 12, pp. 2431–2440, 2012.
- [47] H. Lam, E. Bertini, P. Isenberg, C. Plaisant, and S. Carpendale, “Empirical Studies in Information Visualization: Seven Scenarios,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 9, pp. 1520–1536, 2012.
- [48] S. Jeay, S. Gaulis, S. Ferretti, H. Bitter, M. Ito, T. Valat, M. Murakami, S. Ruetz, D. A. Guthy, C. Rynn, M. R. Jensen, M. Wiesmann, J. Kallen, P. Furet, F. Gessier, P. Holzer, K. Masuya, J. Würthner, E. Halilovic, F. Hofmann, W. R. Sellers, and D. G. Porta, “A distinct p53 target gene set predicts for response to the selective p53–HDM2 inhibitor NVP-CGM097,” *eLife*, vol. 4, p. e06498, 2015. [Online]. Available: <https://elifesciences.org/articles/06498>
- [49] J. Barretina, G. Caponigro, N. Stransky, K. Venkatesan, A. A. Margolin, and et al., “The Cancer Cell Line Encyclopedia enables predictive modelling of anticancer drug sensitivity,” *Nature*, vol. 483, no. 7391, pp. 603–607, 2012. [Online]. Available: <http://www.nature.com/nature/journal/v483/n7391/full/nature11003.html>
- [50] A. D. Cherniack, H. Shen, V. Walter, C. Stewart, B. A. Murray, and et al., “Integrated Molecular Characterization of Uterine Carcinosarcoma,” *Cancer Cell*, vol. 31, no. 3, pp. 411–423, 2017. [Online]. Available: [http://www.cell.com/cancer-cell/abstract/S1535-6108\(17\)30053-3](http://www.cell.com/cancer-cell/abstract/S1535-6108(17)30053-3)
- [51] M. Streit, A. Lex, S. Gratzl, C. Partl, D. Schmalstieg, H. Pfister, P. J. Park, and N. Gehlenborg, “Guided visual exploration of genomic stratifications in cancer,” *Nature Methods*, vol. 11, no. 9, pp. 884–885, 2014.



Katarina Furmanova is a PhD student at the Department of Computer Graphics and Design at the Faculty of Informatics, Masaryk University in Brno, Czech Republic. Her primary research interest is visualization and visual analysis of intermolecular interactions of proteins.



Martin Ennemoser is a PhD student at the Institute of Computer Graphics at the Johannes Kepler University Linz, Austria. His main research field is the visualization of artificial neural network algorithms.



Samuel Gratzl is a Postdoctoral Researcher at the Johannes Kepler University Linz, Austria. He is also a co-founder and CTO of datavisyn. His main research field is guided visual exploration of heterogeneous biomedical data.



Alexander Lex is an Assistant Professor of Computer Science at the Scientific Computing and Imaging Institute and the School of Computing at the University of Utah. Before joining Utah he was a lecturer and a post-doctoral visualization researcher at the Harvard School of Engineering and Applied Sciences. He received his PhD from the Graz University of Technology in 2012. His primary research interests are data visualization, especially applied to molecular biology, and human computer interaction.



Holger Stitz is a PhD student at the Johannes Kepler University Linz Austria in the visualization group of Marc Streit at the Institute of Computer Graphics. His main research interest are multi-attribute time-series data in the context of large networks. For more information see <http://holgerstitz.de>.



Marc Streit is an Associate Professor of Computer Science at the Institute of Computer Graphics at the Johannes Kepler University Linz, Austria. He finished his PhD at the Graz University of Technology in 2011. In 2014 he received a Fulbright scholarship for research and lecturing at the Harvard University. His scientific areas of interest include visualization, visual analytics, and biological data visualization. For more information see <http://marc-streit.com>.



Thomas Zichner is an Associate Principle Scientist at Boehringer Ingelheim RCV GmbH & Co KG in Vienna, Austria. He received his PhD from the European Molecular Biology Laboratory (EMBL) in 2013. His primary research interest is computational cancer genomics.



Miroslava Jaresova is a PhD student at the Department of Computer Graphics and Design at the Faculty of Informatics, Masaryk University in Brno, Czech Republic. She is also a co-founder of NGO Czechitas, its goal is to inspire and empower new talents for stronger diversity and competitiveness in tech. Her primary research interest is security visualization.