Info Vis

Article

Information Visualization
1–23
© The Author(s) 2019
Article reuse guidelines:
sagepub.com/journals-permissions
DOI: 10.1177/1473871619878085
journals.sagepub.com/home/ivi
SAGE

Taggle: Combining overview and details in tabular data visualizations

Katarina Furmanova[1*], Samuel Gratzl[2*], Holger Stitz[3], Thomas Zichner[4], Miroslava Jaresova[5], Alexander Lex[6] and Marc Streit[3]

Abstract

Most tabular data visualization techniques focus on overviews, yet many practical analysis tasks are concerned with investigating individual items of interest. At the same time, relating an item to the rest of a potentially large table is important. In this work, we present Taggle, a tabular visualization technique for exploring and presenting large and complex tables. Taggle takes an item-centric, spreadsheet-like approach, visualizing each row in the source data individually using visual encodings for the cells. At the same time, Taggle introduces data-driven aggregation of data subsets. The aggregation strategy is complemented by interaction methods tailored to answer specific analysis questions, such as sorting based on multiple columns and rich data selection and filtering capabilities. We demonstrate Taggle by a case study conducted by a domain expert on complex genomics data analysis for the purpose of drug discovery.

Keywords

Visualization techniques, tabular data, multidimensional data visualization, aggregation, hierarchical grouping and sorting, degree of interest, focus and context

Introduction

Visualization of tabular or multidimensional data is important in many application domains and is a mainstay of visualization research. Many multidimensional data visualization techniques, however, focus on providing overviews. To answer questions about the high-level similarity of items, projection techniques have proven useful, while exploring correlations between dimensions is well addressed by axes-based techniques such as scatterplot matrices and parallel coordinate plots. The third type of task is concerned with understanding the properties of an item in all dimensions, which is well addressed by tabular techniques. Tabular techniques use a spreadsheet-like layout, with each item in a row and each dimension in a column. In contrast to spreadsheets, the cells use visual encodings to make the data easy to view and to be able to explore higher level trends. Prominent examples of tabular visualization are the Table Lens,[1] Bertifier,[2] LineUp,[3] and ComplexHeatmap.[4]

A shortcoming of current tabular visualization techniques is their lack of sophisticated focus and context.

[1]Masaryk University, Brno, Czech Republic
[2]datavisyn GmbH, Linz, Austria
[3]Johannes Kepler University, Linz, Austria
[4]Boehringer Ingelheim RCV GmbH & Co KG, Vienna, Austria
[5]Czechitas z.s., Prague, Czech Republic
[6]The University of Utah, Salt Lake City, UT, USA

*These authors contributed equally to this work.

Corresponding author:
Marc Streit, Institute of Computer Graphics, Johannes Kepler University Linz, Altenbergerstraße 69, 4020 Linz, Austria.
Email: marc.streit@jku.at

A common solution implemented in both the Table Lens and LineUp is to scale down the rows in the visualization and then use geometric distortion (lenses) to reveal details about selected items. Distortion, however, is associated with a variety of drawbacks, such as maintaining object constancy.[5] Also, lens-based approaches in tables rely on linear orderings, which cannot leverage higher level semantics of the data to provide compact but meaningful aggregations. Aggregation approaches based on grouping, in contrast, can stratify a table in a data-driven way and hence semantically meaningful way.

Our primary contribution is *Taggle*, a tabular visualization method that displays large tabular datasets with up to a million data items by selectively grouping and aggregating subsets of a dataset. The goal of Taggle is to provide a high-level overview of large tabular datasets while allowing users to drill down to individual items. Groupings and aggregations of rows can be dynamically defined by users using selection, or in a data-driven way based on categorical or numerical dimensions. Hierarchical combinations of aggregations enable fine-grained control of what to show in a dataset at which level of detail. Taggle also introduces grouping and aggregation of columns for cases where columns represent data of the same type, as, for example, in time-series data. The grouping and aggregation capabilities are complemented by sorting and filtering techniques.

We showcase Taggle using a public health dataset: the spread of AIDS across the nations of the world. We also demonstrate Taggle using a variety of datasets, including a dataset of soccer players, programming language popularity, world happiness measures, economic data, and many others at https://taggle.caleydoapp.org/. We demonstrate Taggle's utility using a case study on analyzing a cancer genomics dataset for the purpose of drug discovery.

Tabular data

Throughout this article, we use an AIDS dataset from UNAIDS AIDSinfo (https://aidsinfo.unaids.org/) as a guiding example. This dataset was enriched with metadata about the countries, such as population, which we retrieved from the United Nations Population Division (https://www.un.org/en/development/desa/population/) and the yearly Human Development Report of the United Nations Development Programme (http://hdr.undp.org/). The combined dataset consists of 17 numerical columns (e.g. *population*, *sex before the age of 15* in percent), 4 categorical columns (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 or continent, in our example). Lex et al.[6] 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,[7] 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 the age of 15* is stated in percent.

We distinguish between the following data types: Attribute 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, temporal (date and time), or textual. Matrices are composed of attribute columns of the same semantics and data type as is commonly found in, but not exclusive to, time series. An example is a country's *GDP* over multiple years, where each year is a column in the matrix. A non-time-series example, common in the field of genomics, is a gene expression dataset, where the rows are genes and each patient is a column in the matrix. Although it is possible to interpret matrices as a list of columns, it is beneficial to treat them as a matrix, because the homogeneity of the data is an opportunity for compact representation. The columns in matrices can also be associated with attributes that describe a common property of the column, such as the decade associated with a year, or a shared phenotype of patients.

Design goals

Based on discussions with experts from various application domains who regularly analyze large tabular datasets, literature reviews, and our own experience, we developed a set of design goals for Taggle. Our first goal is to develop an item-centric visualization technique that also explicitly shows all dimensions relevant to an analysis task. This goal by itself is addressed by prior tabular data visualization technique, but currently no tabular data visualization technique addresses our second goal: providing a seamless combination of overview and details through selective, data-driven aggregation. A technique that would satisfy this goal would remedy the major drawback of tabular data visualization techniques: limited context. Current tabular visualization techniques can only provide context only by showing neighbors through a single, global sorting, which makes it difficult to compare items of different categories. This design goal is hence concerned with showing the details about selected items and providing context, for example, through aggregations of data-driven groups.

To fully leverage the potential of an overview plus detail tabular data visualization technique, we need to give users the ability to flexibly define the parameters of the display. To address that, our third goal is to provide rich interaction techniques that support answering specific questions, such as sorting, filtering, and grouping. Finally, to appropriately visualize the diverse data types and different levels of aggregations, we need to provide a variety of visual encodings suitable for specific situations. One goal is to provide sensible defaults, but we also need to provide the ability to flexibly choose visual encodings tailored to data types and aggregation levels, to account for the diversity of analysis questions and dataset characteristics.

Related work

We discuss related work in light of two considerations: (1) a review of tabular data visualization techniques and (2) approaches to aggregation.

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 four types of tabular data visualization techniques:

1. Dimensionality reduction techniques, which show a lower dimensional projection of a high-dimensional dataset.
2. Axes-based techniques, which position marks for each cell based on its value, such as parallel coordinates, star plots, and scatterplot matrices,
3. Tabular techniques, which retain item positions across columns and encode the data within the cells.
4. Multiple coordinated view (MCV) and hybrid techniques, which show visualization of individual dimensions or subsets of attributes in separate but linked views.

Our four types of tabular data visualization techniques are related to the three families proposed by Dimara et al.[8] In their work, they distinguish between lossy and lossless geometric projection techniques. Lossy techniques do not preserve the raw values of individual dimensions, that is, this category corresponds to the dimensionality reduction techniques. Their family of lossless techniques includes axes-based and tabular techniques, which we keep separate, as they employ different data encoding principles.

Dimensionality reduction techniques. Projection or dimensionality reduction techniques visualize the structure of items associated with high-dimensional data in a lower dimensional space. There are various commonly used approaches, such as principal component analysis, multidimensional scaling techniques, or t-distributed Stochastic Neighbor Embedding (t-SNE).[9] For data visualization, usually a two-dimensional (2D) or sometimes also a three-dimensional (3D) representation of the projected items is displayed. These low-dimensional projections show groups of similar items close to each other. One problem of projections is that they can produce artifacts showing items that are quite different in proximity. A variety of techniques have been proposed to address this and related shortcomings.[10–12] Another challenge with dimensionality reduction is the sensitivity of the results to the choice of algorithm and the sensitivity to parameters,[13] which often makes an iterative approach with multiple parameters and/or algorithms necessary.

A special case of dimensionality reduction is to turn relationships and items into a network, and then render that network using, for example, force-directed layout algorithms. Examples of this approach are Ploceus,[14] Orion,[15] and Origraph.[16]

We argue that projection techniques are well suited to visualize structure in a high-dimensional dataset, but they cannot adequately show *why* items in a cluster belong together. Projection techniques are especially useful in cases where the dimensions themselves are not meaningful to human analysts, such as a table of term frequencies when analyzing text documents. Taggle is concerned with exactly the opposite use cases: where the properties of the dimensions are critical in making decisions.

Axes-based techniques. Axes-based techniques use axes representing individual attributes and spatially encode the items' values. Key examples are scatterplot matrices,[17,18] which place scatterplots consisting of orthogonal axes to show pairwise relationships between attributes in a matrix, and parallel coordinates,[19–21] which place axes in parallel and connect individual items to their position on the axes using polylines. Variations of parallel coordinates are star plots,[22] where all axes originate from a common point, or other, more general axes-based layouts.[23] Generalizations of axes-based techniques include FLINA,[24] a technique that lets users flexibly arrange axes and choose between connection lines or dots, and GPLOM,[25] which generalizes the scatterplot matrix idea to other visualization techniques shown in the cells.

Axes-based techniques can effectively show correlations between neighboring axes. However, the quality of insights depends on the order of the axes. Other limitations are the visual clutter caused by crossing polylines and the fact that axes-based technique is problematic for encoding categorical and textual attributes.

Tabular techniques. Tabular visualization techniques use a grid layout where rows represent items and columns dimensions (although the inverse is also possible); the value of each item in each dimension is encoded in a cell. Within the class of tabular techniques, we further distinguish tabular visualizations for homogeneous tables, visualizations for heterogeneous tables, and spreadsheet tools. An overview summarizing the features and supported tasks of individual tabular visualization techniques listed in this section can be found in Table 1 of the supplementary material.

The prototypical example of a homogeneous tabular visualization technique is a heatmap,[26] where cell values are encoded using color (hue, saturation, value, or opacity). Homogeneous table visualization tools are useful for data that have the same type and scale across all dimensions (matrices, according to our definition in section "Tabular data"). Heatmaps are exceptionally scalable, as the cells can be allocated as little as a single pixel of space. A key aspect is to find good orderings of the rows and columns, which is often done using clustering[7] or seriation approaches.[27] Visualization tools that provide advanced features for heatmaps include the Hierarchical Cluster Explorer,[28] GAP,[29] PermutMatrix,[30] Clustergrammer,[31] and SmartExplore.[32] Taggle can efficiently visualize homogeneous tables, but in contrast to the techniques discussed here, Taggle also supports heterogeneous tables and can combine homogeneous parts of a heterogeneous table (matrices) and heterogeneous columns in a single visualization. The Table Lens[1] is a tabular visualization technique suitable for heterogeneous tables. It is probably the most closely related technique to Taggle and inspired its development. It uses visual encodings tailored to different data types to represent values in cells. Rich sorting operations allow users to compare trends between separate attributes. Scalability is achieved by down scaling rows, and a combination of appropriately chosen visual encodings and lens techniques ensures the readability of trends and individual items. The most important differences to Taggle are that the Table Lens does not support aggregation and is therefore limited in terms of scalability. Taggle also introduces a variety of subtle new ideas, such as embedding space-efficient techniques for homogeneous subsets of a table. A variety of tools, such as DataComb,[33] the Visual Spreadsheet,[34] and the table views in some multivariate tree and network visualization tools,[35,36] implements ideas of the Table Lens. Another technique employing various visual encodings suitable for heterogeneous tables is Bertifier.[2] It was inspired by Jacques Bertin's matrix analysis methods and supports interactive data reordering based on similarities between rows and columns. However, the technique is intended mainly for presenting small- or medium-sized tables.

Widely used spreadsheet tools, such as Microsoft Excel (https://products.office.com/en-us/excel/), Google Sheets (https://www.google.com/sheets/about/), and Apache OpenOffice Calc (https://www.openoffice.org/product/calc.html) typically support tabular operations such as sorting, filtering, and grouping. However, although spreadsheet tools usually support rich charting operations, they provide only limited support for the direct visual encoding of cells, using techniques such as conditional formatting.

FOCUS[37] and its successor InfoZoom[38] are hybrid spreadsheet/tabular visualization tools. In addition to the Table-Lens-like layout, InfoZoom provides an overview mode that shows the distribution of values for individual attributes, sorting each attribute row individually. Although this provides an overview of the distribution of values, it is no longer a tabular layout.

MCV techniques and hybrids. MCV systems represent (sets of) attributes of a tabular dataset in separate, linked views. 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 highlight the same items in different views. Representative systems in this category include Improvise[39] and Keshif.[40] Common configurations of Keshif, for example, use a tabular view to identify specific items, but represent other attributes in other views using histograms or bar charts, for instance.

Although 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.[41] 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: although 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.

Note that the line between MCVs and other techniques is fluid; a scatterplot matrix, for example, can be considered as both an axes-based technique and an MCV system.

Hybrid approaches that use multiple views and combine overview and tabular approaches or overview and projection approaches are also available. 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,[42] VisBricks,[6] StratomeX,[43,44] Domino,[45] and Furby.[46] 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.[11] Domino[45] is a hybrid tabular/overview MCV technique. It is based on the concept of placing subsets of a dataset on canvas and choosing a suitable representation (view) for it. Multiple subsets can then be connected to show their relationships in various ways. Matchmaker,[47] VisBricks,[6] and StratomeX[43,44] are related hybrid techniques but they are more restricted with respect to the selection and layout of subsets.

Aggregation methods

Orthogonal to the design space discussed above are aggregation methods for tabular data: 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. Elmqvist and Fekete[48] 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.

Examples of overview techniques using aggregation are hierarchical parallel coordinates,[49] which visualize cluster centroids rather than individual items, and VisBricks,[6] which can visualize clusters using various techniques, including statistical summaries such as histograms. An example MCV technique that predominantly uses aggregations is Keshif.[40] In Keshif, 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. Although 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.

Various specialized tabular visualization tools use aggregation in tabular layouts. iHAT[50] 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.[51] 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[52] aggregates rows or columns of a table based on a pre-existing aggregation hierarchy. Users can traverse the hierarchy and pivot through intersecting hierarchies. The UpSet[53] technique aggregates items based on set memberships. It uses visualizations such as box plots for representing aggregated group statistics. In contrast to these techniques, Taggle provides the user with the flexibility to aggregate subsets of the table, while keeping details of other parts of the table visible in place.

Visualization and interaction design

Taggle is an item-centric visualization technique that shows all dimensions relevant to an analysis task and at the same time provides a seamless combination of overview and details through selective, data-driven aggregation. Here we introduce this approach.

Taggle enables users to group items based on hierarchical combinations of attributes. The result of these nested grouping levels is an ordered tree where all leaves are items (Figure 1(a)). Data-driven filter and sorting operations (Figure 1(b) and (c)) can be used to reveal items of interest.

By defining groups, we can add new levels to the tree (Figure 1(d) and (e)). For example, we can group the countries in the AIDS dataset by continent. Groups can be defined based on categorical attributes, numerical thresholds, or user selections. Groups are represented as a row showing summary representations for the items in the group.

Each branch in the tree can be collapsed independently, hiding the items while the group summary remains, as shown in Figure 1(f). Each row of the resulting table then corresponds to either one item or one group. We can use this approach, for example, to show summaries of all continents, but also to show the individual countries on the African continent. By adjusting the level at which to aggregate, users can dynamically control the level of detail of the rows when rendering the table.[48]

Finally, we introduce a degree of interest operation[54] to reveal aggregated items that are especially relevant to the analysis. Our current implementation is naive, revealing only the first N items of an aggregated group. By leveraging sorting, we ensure that these items are the most relevant to the current analysis task. The operation allows us, for example, to show a summary of the AIDS epidemic by continent and reveal the 10 most affected countries for each continent at the same time. The degree of interest can be adjusted to reveal more or fewer items (Figure 1(g)). This function could be improved to take other aspects of the data into account, such as a cut-off of an attribute or the size of the group.

Overall design

The Taggle interface consists of two parts, as shown in Figure 2: (a) the main table view and (b) a data selection panel that is the interface for various operations. The table view implements the overview plus detail concept for visualizing tabular data. The column headers of the table view provide the means for sorting, changing visual encoding, filtering, and grouping. The data selection panel provides access to all available numerical, categorical, text, and matrix attributes. Its primary use is to enable analysts to choose which attributes to show in the table view. For each column that is shown in the table view, the data selection panel shows a visual summary of the data in the form of a histogram, when appropriate. Below, we introduce the visual elements and interactions in detail, together with justifications of our design decisions.

Layout strategy

Complementary to our overview plus detail concept described above, we introduce two different layout modes serving the high-level tasks of (1) obtaining an overview and (2) seeing details for a subset of the items.

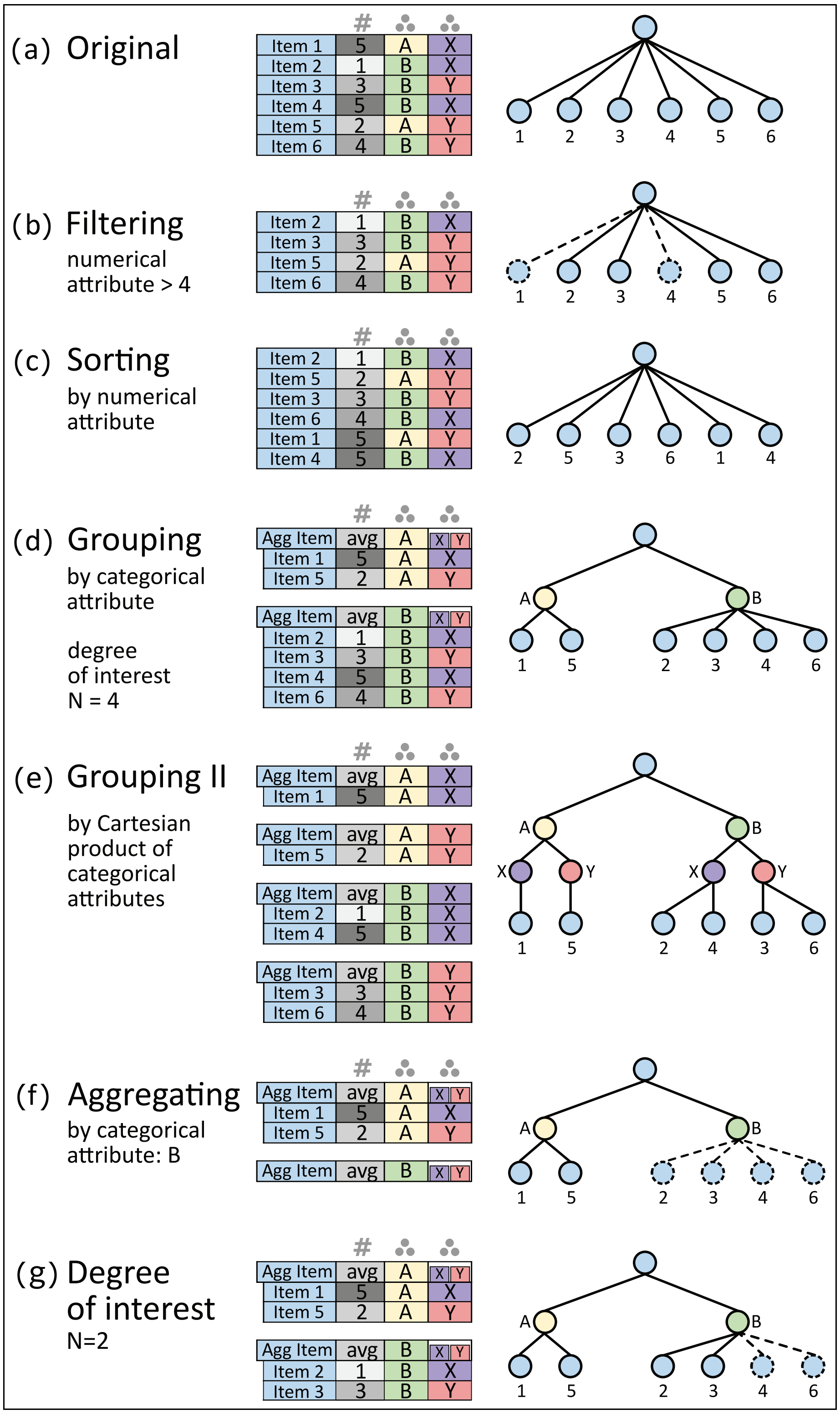
The goal of the detail mode is to allow users to see all details for selected items including labels, numerical values, and category names. Although this maximizes the readability of items, it comes at the cost of reducing the number of visible items.

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. To achieve this, Taggle decreases the height of items until the whole table fits on the screen, or until each item has a height of a single pixel, as lower values would introduce uncertainty due to interpolation artifacts.[51] Aggregated groups are shown using a fixed height. Overview mode is a complementary strategy to aggregation: it is useful to get an idea about the distribution of the data in the columns and does not require that meaningful groups are defined. When viewing the table in overview mode, users can still increase the level of detail for one or multiple items by selecting them, which is useful in cases where users spot items of interest that they want to inspect in detail.

Sorting

Sorting is a simple way of identifying minima and maxima in columns. Sorting also reveals relationships between columns. In addition to sorting in ascending or descending order by a numerical, textual, or categorical column, 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 sorting strategy 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 the sorting criterion.

Although other table visualizations such as the Visual Spreadsheet[55] sort attributes hierarchically based on the order of the columns, we decided to separate the sorting from the layout. Since we expect that in most cases users are satisfied with simple sorting by one attribute, clicking on the sort button in the column header always results in the data being sorted by the corresponding attribute. Once the user activates the sorting by one attribute, a dedicated sorting hierarchy panel appears in the data selection panel. The



(a)

	#	∴	∴
Item 1	5	A	X
Item 2	1	B	X
Item 3	3	B	Y
Item 4	5	B	X
Item 5	2	A	Y
Item 6	4	B	Y

(b)

	#	∴	∴
Item 2	1	B	X
Item 3	3	B	Y
Item 5	2	A	Y
Item 6	4	B	Y

(c)

	#	∴	∴
Item 2	1	B	X
Item 5	2	A	Y
Item 3	3	B	Y
Item 6	4	B	X
Item 1	5	A	Y
Item 4	5	B	X

(d)

	#	∴	∴
Agg Item	avg	A	X Y
Item 1	5	A	X
Item 5	2	A	Y
Agg Item	avg	B	X Y
Item 2	1	B	X
Item 3	3	B	Y
Item 4	5	B	X
Item 6	4	B	Y

(e)

	#	∴	∴
Agg Item	avg	A	X
Item 1	5	A	X
Agg Item	avg	A	Y
Item 5	2	A	Y
Agg Item	avg	B	X
Item 2	1	B	X
Item 4	5	B	X
Agg Item	avg	B	Y
Item 3	3	B	Y
Item 6	4	B	Y

(f)

	#	∴	∴
Agg Item	avg	A	X Y
Item 1	5	A	X
Item 5	2	A	Y
Agg Item	avg	B	X Y

(g)

	#	∴	∴
Agg Item	avg	A	X Y
Item 1	5	A	X
Item 5	2	A	Y
Agg Item	avg	B	X Y
Item 2	1	B	X
Item 3	3	B	Y

Figure 1. Illustration of topological operations on a heterogeneous table (a) consisting of numerical (#) and categorical (∴) attributes and their results reflected in the aggregation hierarchy: (b) filtering, (c) sorting, (d) grouping by a single categorical attribute, (e) grouping by the Cartesian product of two categorical attributes, (f) aggregating, and (g) degree of interest.

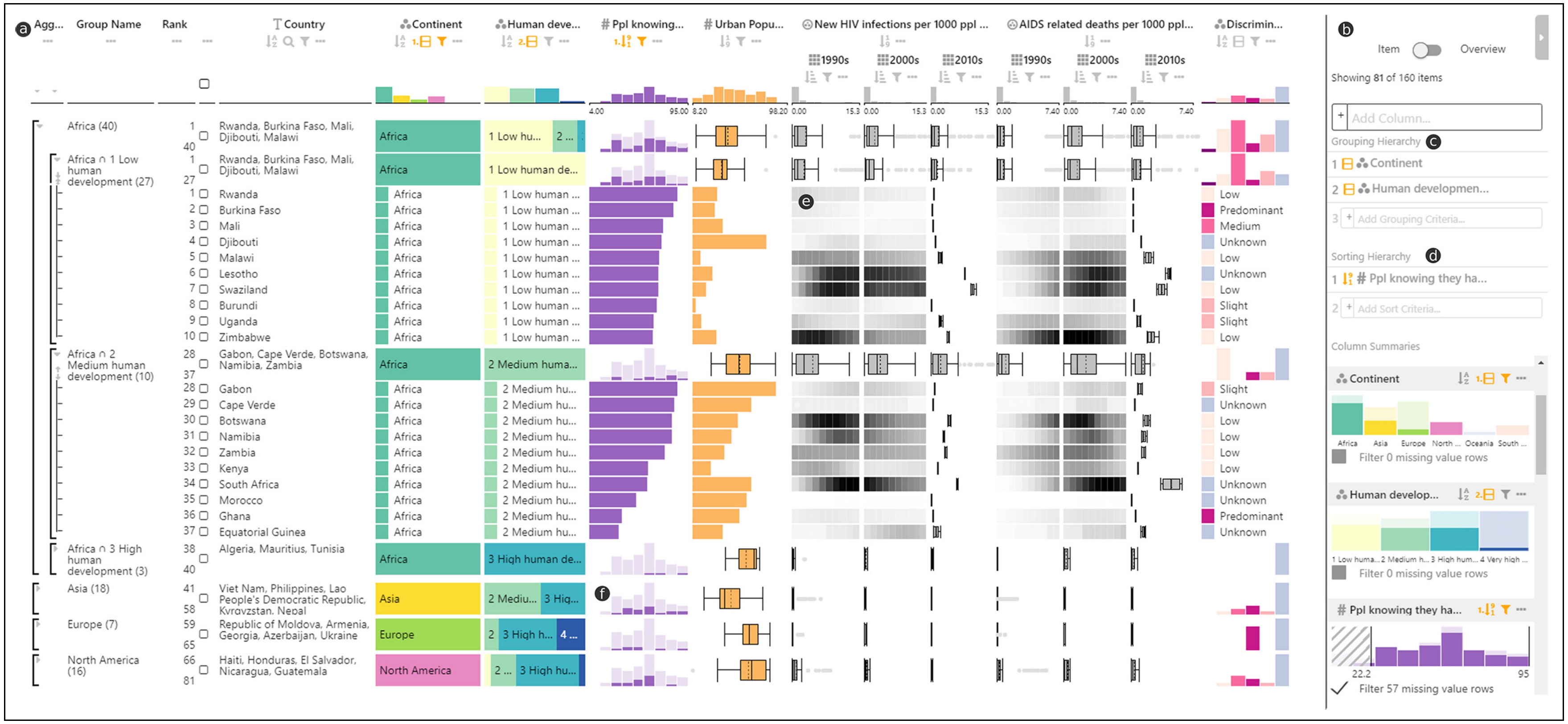


Figure 2. The Taggle interface consisting of (a) a table view and (b) a data selection panel showing a dataset on AIDS in several countries grouped by continent and level of human development index. The data selection panel consists of (c) grouping and (d) sorting hierarchy panels and attribute filter views that allow users to filter out records by interacting with the histograms. The rows with individual African countries indicate the relationship between *new HIV infections* and *AIDS-related deaths* over time. (e) 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. (f) The rows of countries in Asia, Europe, and North America have been aggregated to histograms, box plots, and stacked bars.

panel allows users to add additional sorting attributes and change their order (see Figure 2(d)).

Filtering

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 2(b), *people knowing they have HIV*) or by selecting categories that are to be removed from the table (Figure 2(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. As an alternative to setting filters in the data selection panel, users can open a filter dialog via the header of the columns.

Grouping and aggregation

Being able to stratify tables into meaningful groups is not only an important feature for structuring tabular data, but also an essential prerequisite for aggregation operations in Taggle. Grouping is related to sorting since grouping also influences the order of items. Taggle, however, separates these operations in order to enable more fine-grained control of groups. As discussed before, we leverage categorical or binned numerical attributes to group datasets. Similarly, we can leverage regular expressions on string columns to create groups, or use dates and date ranges on date columns. Users can also split the table into two groups based on the current selection. Combining multiple hierarchically sorted grouping attributes creates fine-grained groups that correspond to the Cartesian product of the constituting categories. In practice, we found that two to three grouping levels are sufficient, because more lead to fragmented groups.

Setting the grouping hierarchy is analogous to hierarchical sorting—the order of grouping attributes is indicated in a dedicated grouping hierarchy panel. Since grouping takes precedence over sorting, the hierarchy is shown above the sorting hierarchy panel (see Figure 2(c)). The separation of grouping and sorting operations gives the user tighter control over the order of the table items. Users can, for example, group the items based on a binned numerical attribute but sort the items inside the groups according to a different attribute.

A group name column summarizes the current grouping and how many items are contained. In Figure 2, for instance, the combination of the attributes *continent* and *the human development index* constitute the grouping, which is indicated in the first column. Groups can also be sorted by their name, by the number of contained items, by statistical measures of numerical attributes (e.g. mean or median), or by the most frequent category. Selected options are shown in an additional group sorting hierarchy in the panel.

Figure 3 illustrates a case in which the countries were first grouped based on the percentage of women who had *sex before the age of 15* with a threshold set to 15% (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 the age of 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*.

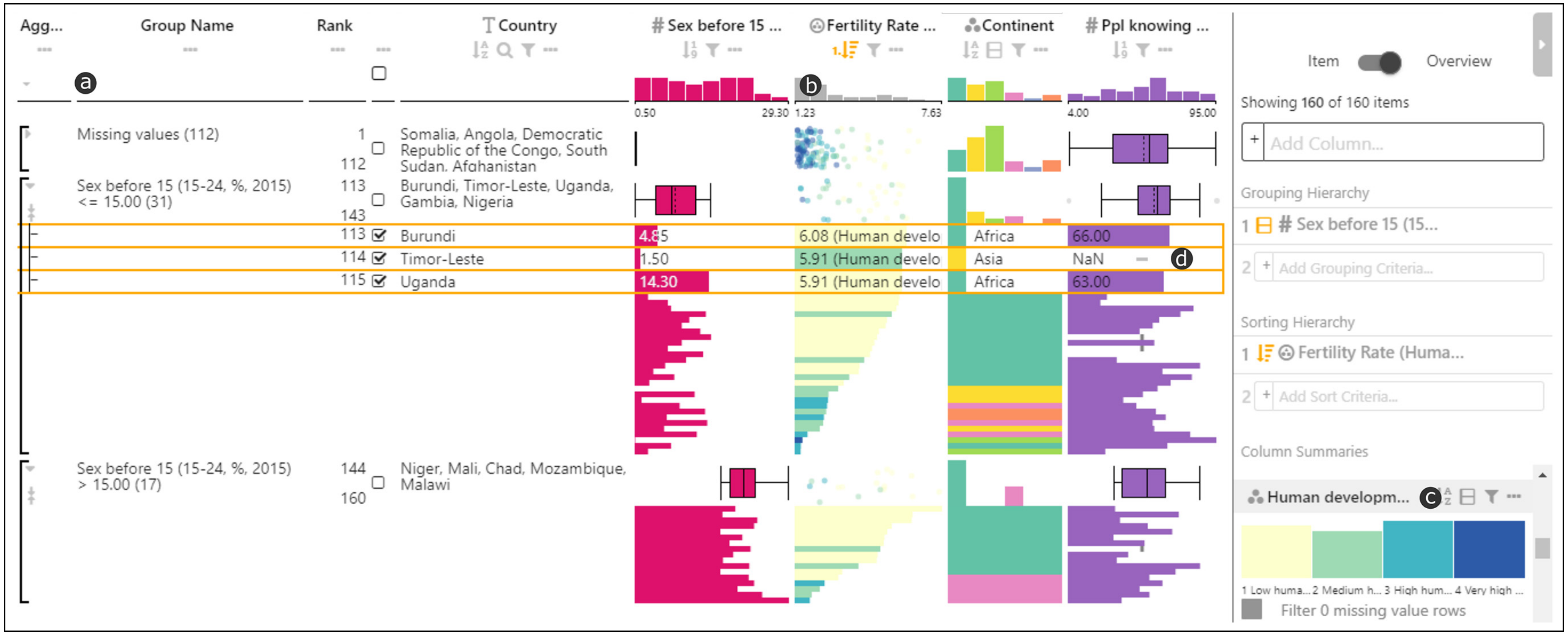
Groups are represented by rows showing an aggregate of the items they contain. Group headers are assigned a uniform height that is about twice that of a row shown in detail mode. We use dedicated visual encodings for aggregate items. For example, instead of bar plots for individual items, we show a histogram or a box plot that represents the whole group (see Figure 2(f)). As discussed earlier, the items in a group can be shown below its header, partially hidden based on a degree of interest function, or completely hidden. In Figure 2, for instance, only the first 10 African countries with low and medium *human development index* are displayed.

Visualizing matrices

While many tools offer support for time-series data (e.g. by showing sparklines), these tools usually do not support general matrices. For example, the option to reorder the data points is usually missing, because it is not necessary for the time-series data. In our technique, adding a matrix to a table visualization introduces a second key for the columns of the matrix. 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 1000 people* matrix and the *AIDS-related deaths per 1000 people* matrix in Figure 2(e) 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.

Encoding and multiform visualizations

The table view encodes each selected column or matrix using one of multiple alternative visual encodings suitable for the data type, including bars, dots,


Agg...
Group Name
Rank
Country
# Sex before 15 ...
Fertility Rate ...
Continent
# Ppl knowing ...
a
b
c
d
0.50
29.30
1.23
7.63
4.00
95.00
Missing values (112)
Sex before 15 (15-24, %, 2015) <= 15.00 (31)
Sex before 15 (15-24, %, 2015) > 15.00 (17)
Somalia, Angola, Democratic Republic of the Congo, South Sudan, Afghanistan
Burundi, Timor-Leste, Uganda, Gambia, Nigeria
Niger, Mali, Chad, Mozambique, Malawi
Item
Overview
Showing 160 of 160 items
Add Column...
Grouping Hierarchy
1 # Sex before 15 (15...
2 Add Grouping Criteria...
Sorting Hierarchy
1 Fertility Rate (Huma...
2 Add Sort Criteria...
Column Summaries
Human developm...
1 Low huma... 2 Medium h... 3 High hum... 4 Very high ...
Filter 0 missing value rows

Rank		Country	Sex before 15	Fertility Rate	Continent	Ppl knowing
113	✓	Burundi	4.85	6.08 (Human develo	Africa	66.00
114	✓	Timor-Leste	1.50	5.91 (Human develo	Asia	NaN
115	✓	Uganda	14.30	5.91 (Human develo	Africa	63.00

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

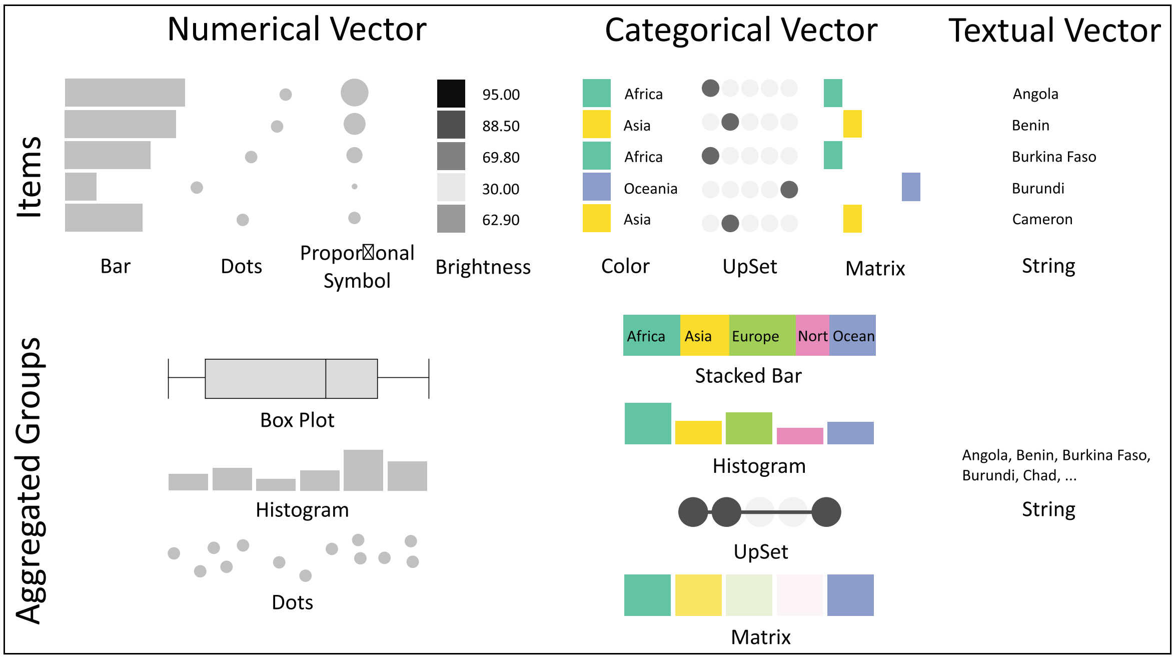


Figure 4. Attribute column visualization techniques for items and aggregated groups by data type. Numerical items can be encoded with bars, dot plots, proportional symbols, or brightness. For categorical items, we offer color encoding plus labels, and two variants of matrix representations, one with and one without color used redundantly. All items can also be displayed as strings. Numerical attributes can be aggregated into box plots and histograms. Distributions of categorical values can be shown as a histogram, a stacked bar, a binary presence/absence matrix inspired by UpSet,[53] or an aggregated matrix with brightness encoding the frequency of individual categories in the group. An aggregated textual attribute shows examples of the group members.

proportional symbols, or brightness for numerical data; color or positional/matrix encoding for categorical data; and heatmaps for matrices.

Following the multiform principle,[6] the visual encoding for each column can be changed on demand. 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 textual 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. In theory, the aggregation choices for the matrix rows and columns should be symmetric. Our limited choices of visualizations for aggregated rows and columns stem from our design decision to show the aggregated rows with fixed height, whereas for aggregated columns the width is flexible and by default reflects the width of the matrix. Thus, most of the visualizations available for aggregated columns (e.g. box plots or dot plots) are not suitable for aggregated rows, as there would not be sufficient space.

We limit ourselves to these choices because they either offer good perceptual properties (e.g. encoding by position) or are very compact, thus allowing users to choose between 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 attribute.[56]

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), but are still clearly visible at all levels of detail.

Compact encodings. When the height of rows is reduced in overview mode, we take various measures to adapt the visualization to the diminished space. 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

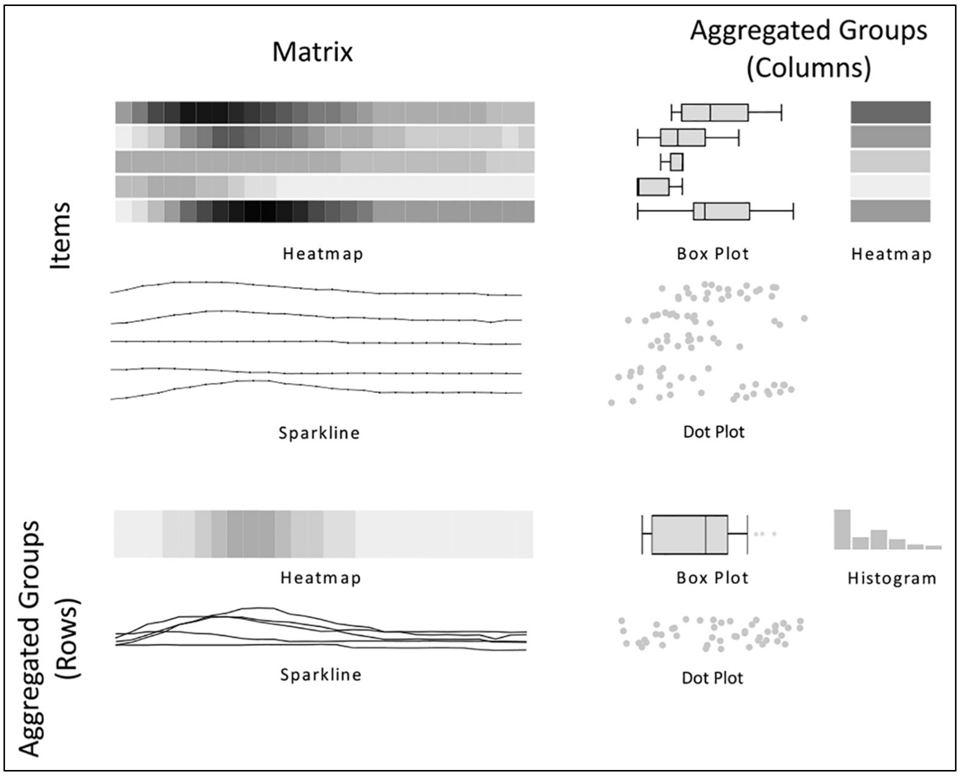


Figure 5. Matrix visualization techniques. Matrix items can be encoded using brightness and as sparklines. Matrices can be aggregated in both column and row directions. When a matrix is aggregated in the column direction, a group of matrix columns within one row is merged into a single cell. The aggregated values can then be visualized using box plots, dot plots, and heatmap. When a matrix is aggregated in the 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.

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 downscaled version. We do not render such cells in overview mode. Examples of visual compression for individual visualization options can be found in Figure 6.

Animated transitions

We support users in understanding changes in the visualization by applying animated transitions,[57] as demonstrated in the accompanying video. Our implementation incorporates smooth transitions for the switch between overview and detail 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.[57] 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 animation is designed to help users understand why rows outside the viewport become visible at the bottom of the table. Similarly, when items in a group are collapsed, we first fade out the items and then gradually reduce the height of the empty group area to the fixed height of the aggregated group.

Combining columns

Giving users the ability to flexibly combine columns supports various tasks. Users can interactively create combined columns by dragging either existing ones on an empty container or 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, as shown in Figure 7(a) and (e). It encloses multiple individual columns by adding a joint header above all columns contained. Nested multiple columns are useful for creating semantic groups. The nested column 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[3,58] by combining two or multiple numerical

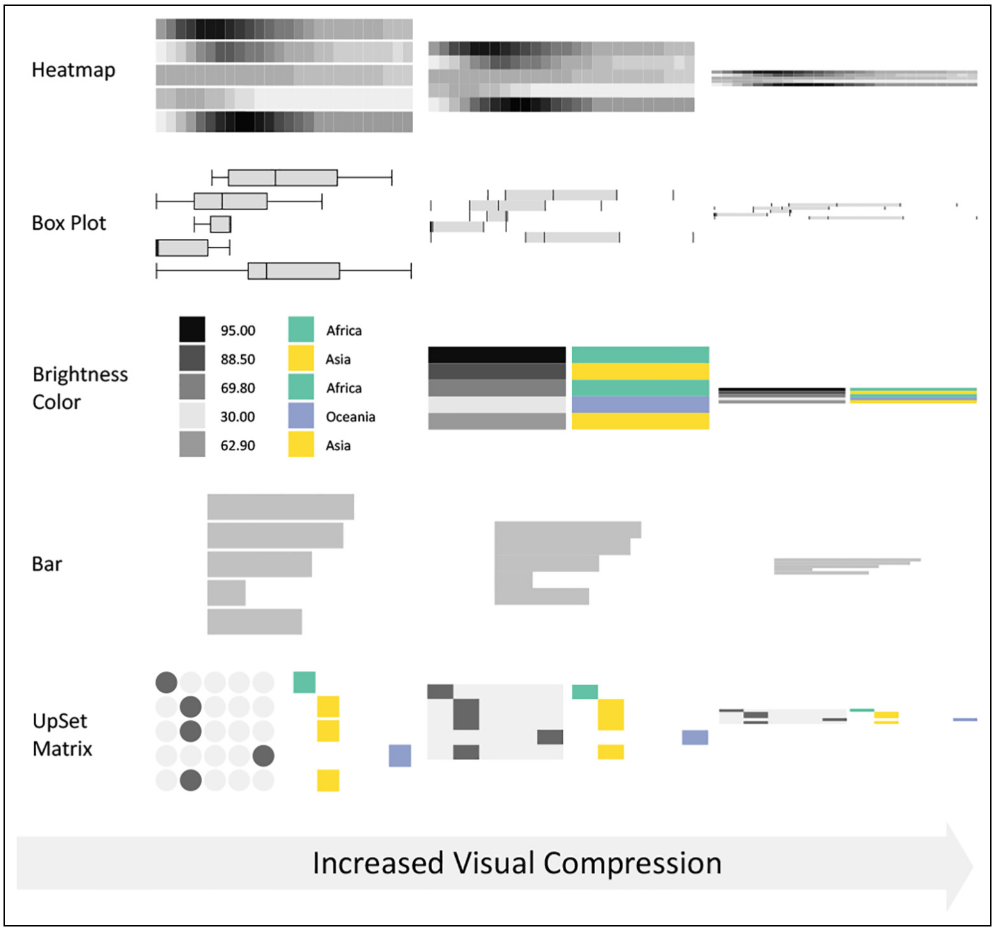


Figure 6. Example of encodings at different scales. In the first column, the items are displayed at full height with white space separating the rows. If a textual label is part of the visualization, it is displayed at a readable size. Compact representations (columns two and three) remove white space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

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 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 from bars or dots, or, in case an aggregate is interleaved, from a box plot.

With imposition columns, users can color the visual marks (bar, proportional symbol, etc.) of a numerical column by the color coding of a categorical attribute, 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 attributes into a single numerical column. In addition, users can add scripted columns that allow them to define their own functions via a scripting interface.[3]

Sorting and grouping of column subsets

Although 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[59] or bands for showing relationships between aggregated groups,[43,45] defacto enabling users to create hybrid tabular/overview representations (see Figure 8), and in the

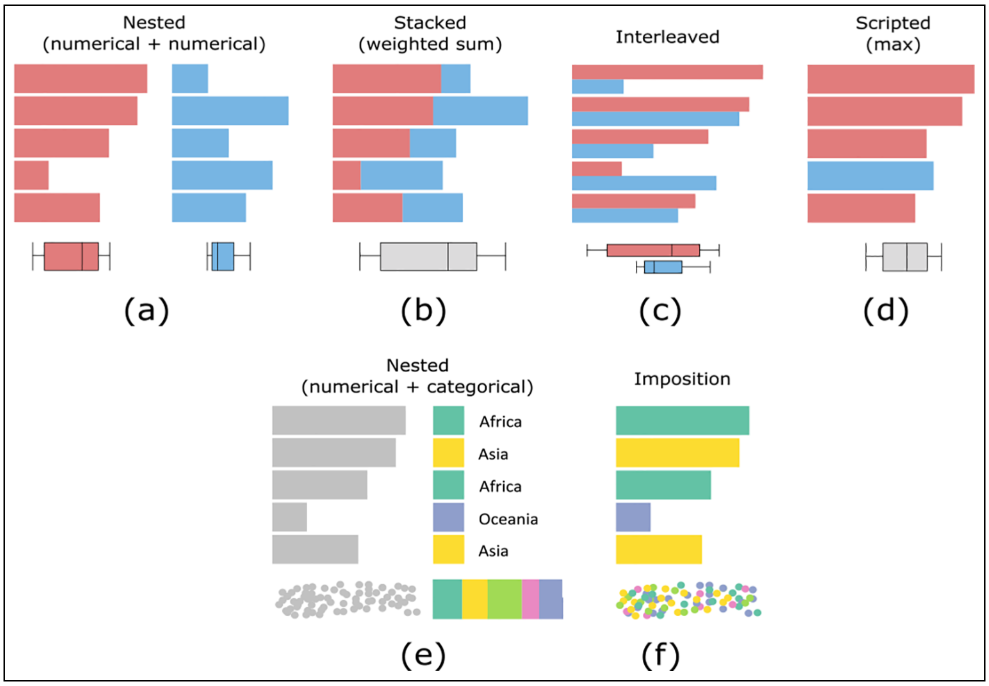


Figure 7. Possible column combinations: (a, e) nested column that semantically groups columns of various types; (b) stacked column that creates a stacked bar plot based on multiple weighted numerical columns; (c) interleaved column that stacks the visualizations of multiple numerical columns; (d) scripted 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.

extreme, even visualization techniques such as parallel sets.[60]

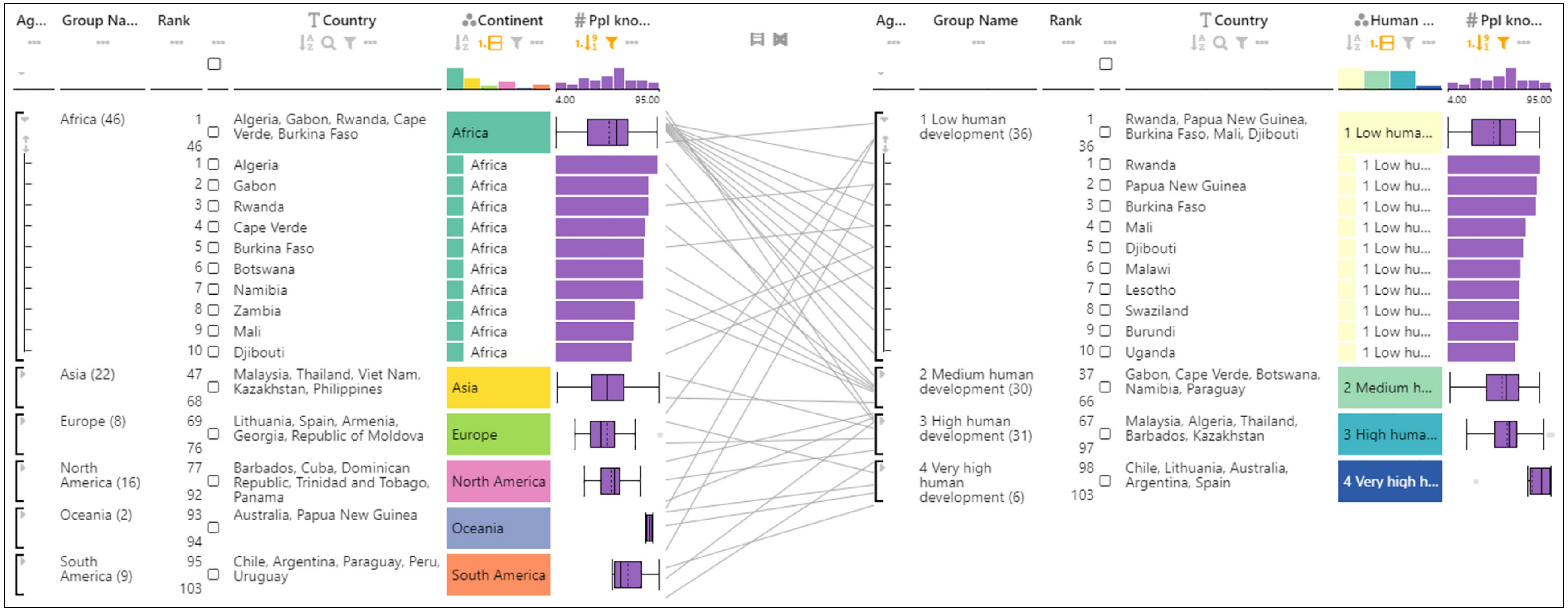
Implementation

In the demo application (https://taggle.caleydoapp.org/), users can switch between multiple preloaded datasets, upload new, and download existing datasets in various formats. Users can locally save and restore a Taggle table together with the analysis session that includes the history of all user interactions.

The Taggle feature set is fully integrated into the *LineUp.js* library (https://lineup.js.org/), which is written in TypeScript and available as open source (https://github.com/lineupjs/lineupjs/). A demo version can be accessed at https://taggle.caleydoapp.org/. Making Taggle available as an open-source library increases the potential for adoption of the technique. Taggle is also designed to be combined with other techniques. To that end, we provide various interfaces. For example, the library can be embedded in Jupyter Notebooks (https://jupyter.org/), (https://github.com/lineupjs/lineup_widget/) and used as an HTML widget (https://www.htmlwidgets.org/), (https://github.com/lineupjs/lineup_htmlwidget), which allows integration into Shiny applications (https://shiny.rstudio.com/), R Notebooks (https://rmarkdown.rstudio.com/r_notebooks.html), Anuglar.js (https://angularjs.org/), Vue.js (https://vuejs.org/), and React.js (https://reactjs.org/). We provide examples for how to embed Taggle in each of these frameworks in the repository. Note that Taggle can also be embedded as a component inside a larger web-based application. The case study described in the following section is based on the Ordino visual cancer analysis tool.[61] The server-side of Ordino retrieves over 500 GB of cancer data from a PostgreSQL database. Complex aggregation queries that need to iterate over a large set of table entries are handled by the database, while the client side with Taggle then receives only the data subset needed for rendering.

Case study: drug target discovery

Taggle was developed in tight collaboration with domain experts working on a drug discovery team at a pharmaceutical company. We demonstrate Taggle by means of a case study conducted on complex genomics data for the purpose of drug target discovery. The case study summarizes an analysis session carried out by one of our collaborators. For the case study, we integrated Taggle into the Ordino Target Discovery Platform[61] that provides access to the required cancer



Group Na...	Rank	Country	Continent
Africa (46)	1–46	Algeria, Gabon, Rwanda, Cape Verde, Burkina Faso	Africa
	1	Algeria	Africa
	2	Gabon	Africa
	3	Rwanda	Africa
	4	Cape Verde	Africa
	5	Burkina Faso	Africa
	6	Botswana	Africa
	7	Namibia	Africa
	8	Zambia	Africa
	9	Mali	Africa
	10	Djibouti	Africa
Asia (22)	47–68	Malaysia, Thailand, Viet Nam, Kazakhstan, Philippines	Asia
Europe (8)	69–76	Lithuania, Spain, Armenia, Georgia, Republic of Moldova	Europe
North America (16)	77–92	Barbados, Cuba, Dominican Republic, Trinidad and Tobago, Panama	North America
Oceania (2)	93–94	Australia, Papua New Guinea	Oceania
South America (9)	95–103	Chile, Argentina, Paraguay, Peru, Uruguay	South America

Group Name	Rank	Country	Human ...
1 Low human development (36)	1–36	Rwanda, Papua New Guinea, Burkina Faso, Mali, Djibouti	1 Low huma...
	1	Rwanda	1 Low hu...
	2	Papua New Guinea	1 Low hu...
	3	Burkina Faso	1 Low hu...
	4	Mali	1 Low hu...
	5	Djibouti	1 Low hu...
	6	Malawi	1 Low hu...
	7	Lesotho	1 Low hu...
	8	Swaziland	1 Low hu...
	9	Burundi	1 Low hu...
	10	Uganda	1 Low hu...
2 Medium human development (30)	37–66	Gabon, Cape Verde, Botswana, Namibia, Paraguay	2 Medium h...
3 High human development (31)	67–97	Malaysia, Algeria, Thailand, Barbados, Kazakhstan	3 High huma...
4 Very high human development (6)	98–103	Chile, Lithuania, Australia, Argentina, Spain	4 Very high h...

Figure 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 the *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.

genomics data (https://ordino.caleydoapp.org/). Note that the collaborator has experience with using interactive visualization tools and was involved in all phases of the project and provided continuous feedback during the development. For the case study, the domain expert operated Taggle himself without the help of visualization experts.

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 status. The expression level of these genes can hence be used to predict the mutation status of *TP53*.[62]

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 1009 cell lines from the public CCLE dataset[63] into Taggle. By default, the table contains a textual column representing the *names of cell lines* and a categorical column indicating *tumor type*. Since only a subset of tumor types is of interest, the analyst filters for *astrocytoma/glioblastoma* (type of cancer of the brain), *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 a 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 observation by also stratifying by mutation status and aggregating all groups (see Figure 10). The expression box plots show good separation for *astrocytoma/glioblastoma* and *melanoma*, whereas 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 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 these cell lines fulfill the analyst's requirements.

Expert feedback

Our collaborators initially planted the seed that led to the development of Taggle by pointing out restrictions they face in current drug discovery tools. They particularly mentioned the need of seamlessly combining overview and details in tabular data analysis for drug discovery.

During the conception and development of Taggle, we had biweekly feedback sessions and in-depth discussions with our collaborators on every aspect of both the concept and the visual interface. The most critical feedback on early prototypes was about the limited rendering performance that hindered their use in real-world scenarios. After making the prototypes more scalable, we received valuable and very detailed feedback on a conceptual level but also regarding the usability of the prototype implementation. For example, the user interface workflow and visual encoding of

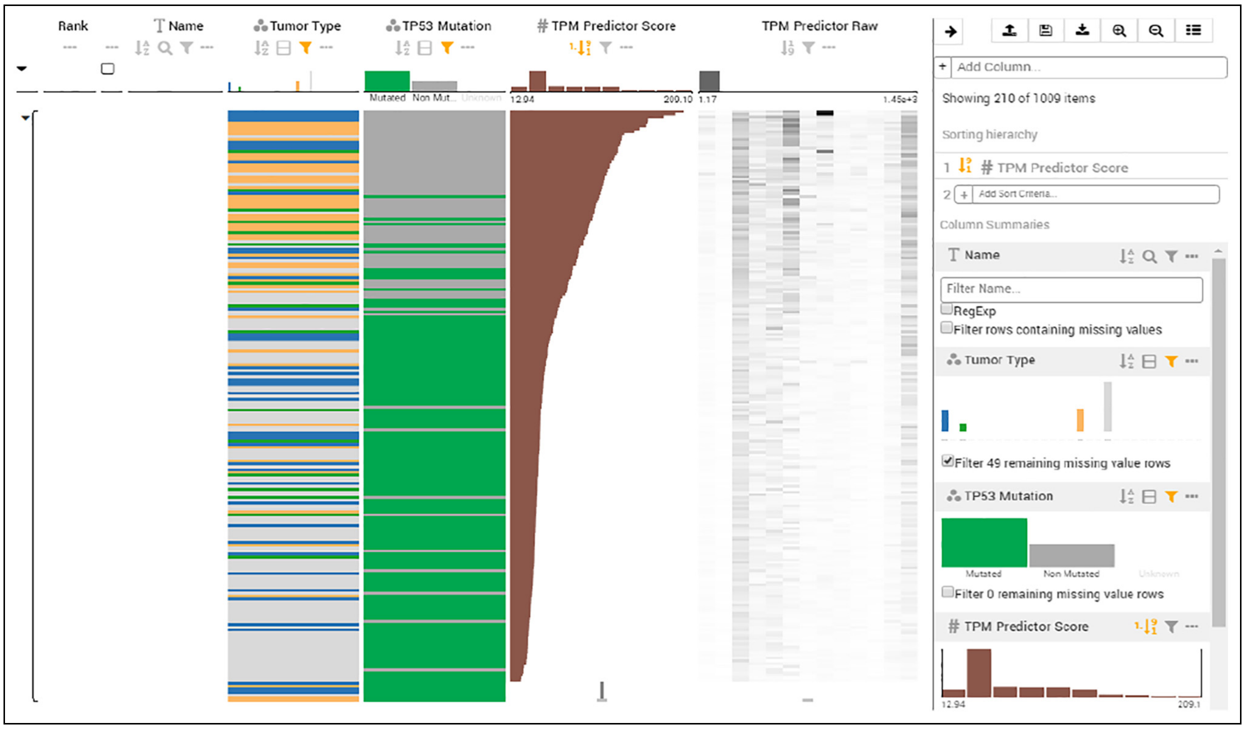
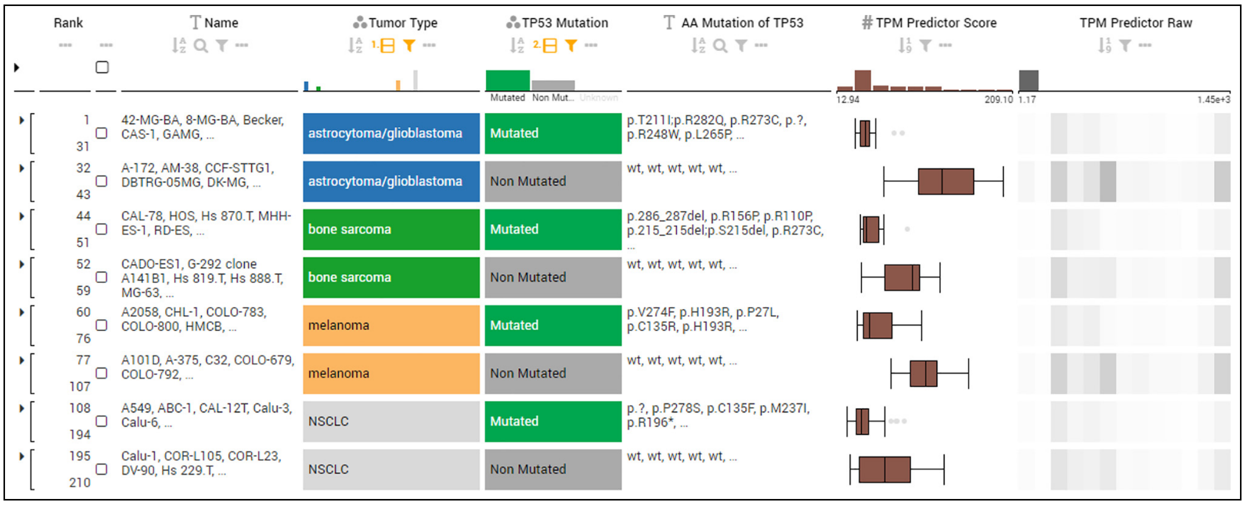


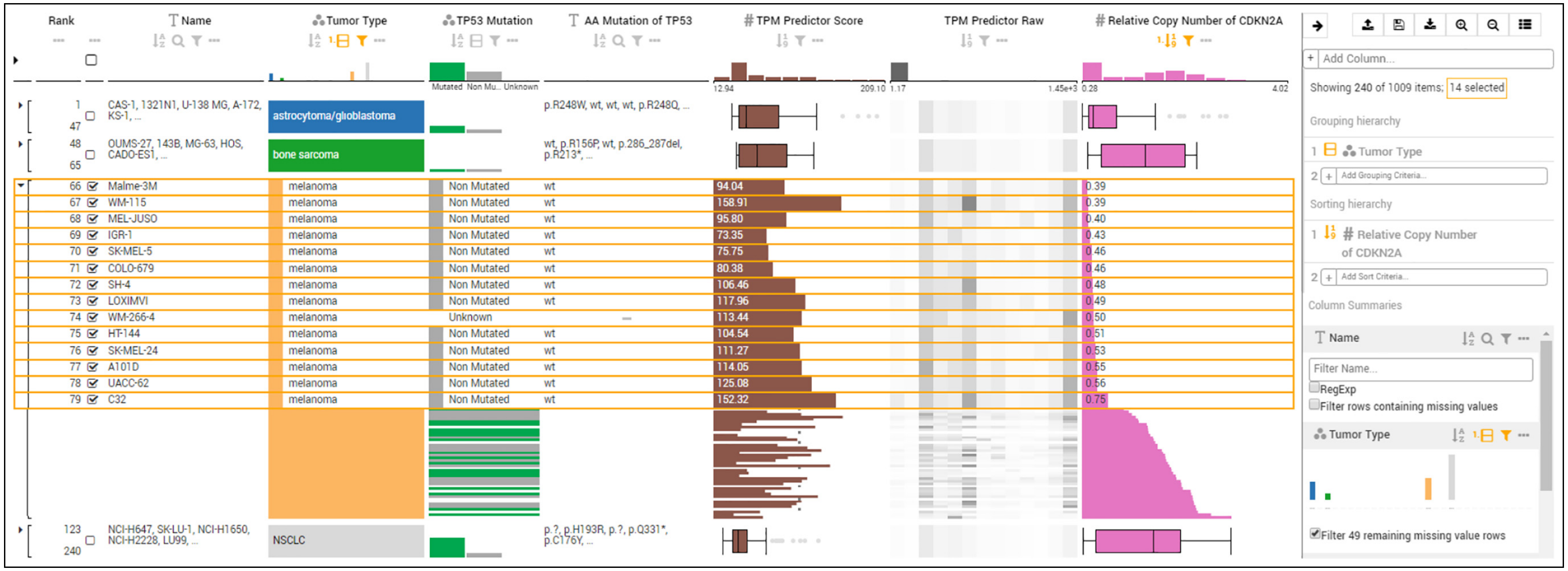
Figure 9. After sorting the cell lines by the *TP53 predictor score* (brown), the analyst notices that those with a low average score are much more likely to be mutated (green). From this observation, the analyst concludes that predicting the mutation status based on the average expression of the 13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



Rank	Name	Tumor Type	TP53 Mutation	AA Mutation of TP53	TPM Predictor Score	TPM Predictor Raw
1–31	42-MG-BA, 8-MG-BA, Becker, CAS-1, GAMG, ...	astrocytoma/glioblastoma	Mutated	p.T211I;p.R282Q, p.R273C, p.?, p.R248W, p.L265P, ...		
32–43	A-172, AM-38, CCF-STTG1, DBTRG-05MG, DK-MG, ...	astrocytoma/glioblastoma	Non Mutated	wt, wt, wt, wt, wt, ...		
44–51	CAL-78, HOS, Hs 870.T, MHH-ES-1, RD-ES, ...	bone sarcoma	Mutated	p.286_287del, p.R156P, p.R110P, p.215_215del;p.S215del, p.R273C, ...		
52–59	CADO-ES1, G-292 clone A141B1, Hs 819.T, Hs 888.T, MG-63, ...	bone sarcoma	Non Mutated	wt, wt, wt, wt, wt, ...		
60–76	A2058, CHL-1, COLO-783, COLO-800, HMCB, ...	melanoma	Mutated	p.V274F, p.H193R, p.P27L, p.C135R, p.H193R, ...		
77–107	A101D, A-375, C32, COLO-679, COLO-792, ...	melanoma	Non Mutated	wt, wt, wt, wt, wt, ...		
108–194	A549, ABC-1, CAL-12T, Calu-3, Calu-6, ...	NSCLC	Mutated	p.?, p.P278S, p.C135F, p.M237I, p.R196*, ...		
195–210	Calu-1, COR-L105, COR-L23, DV-90, Hs 229.T, ...	NSCLC	Non Mutated	wt, wt, wt, wt, wt, ...		

Figure 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.

the hierarchical grouping and sorting capabilities led to confusion. We resolved this problem by introducing an explicit sorting and grouping hierarchy in the data selection panel (see sections “Sorting” and “Grouping and aggregation”). Based on follow-up feedback, we also added the capability of controlling the order of groups, to sort them by number of items or by group name, for instance. The fact that Taggle recently



Rank	Name	Tumor Type	TP53 Mutation	AA Mutation of TP53	TPM Predictor Score	TPM Predictor Raw	Relative Copy Number of CDKN2A
1 – 47	CAS-1, 1321N1, U-138 MG, A-172, KS-1, ...	astrocytoma/glioblastoma		p.R248W, wt, wt, wt, p.R248Q, ...			
48 – 65	OUMS-27, 143B, MG-63, HOS, CADO-ES1, ...	bone sarcoma		wt, p.R156P, wt, p.286_287del, p.R213*, ...			
66	Malme-3M	melanoma	Non Mutated	wt	94.04		0.39
67	WM-115	melanoma	Non Mutated	wt	158.91		0.39
68	MEL-JUSO	melanoma	Non Mutated	wt	95.80		0.40
69	IGR-1	melanoma	Non Mutated	wt	73.35		0.43
70	SK-MEL-5	melanoma	Non Mutated	wt	75.75		0.46
71	COLO-679	melanoma	Non Mutated	wt	80.38		0.46
72	SH-4	melanoma	Non Mutated	wt	106.46		0.48
73	LOXIMVI	melanoma	Non Mutated	wt	117.96		0.49
74	WM-266-4	melanoma	Unknown	–	113.44		0.50
75	HT-144	melanoma	Non Mutated	wt	104.54		0.51
76	SK-MEL-24	melanoma	Non Mutated	wt	111.27		0.53
77	A101D	melanoma	Non Mutated	wt	114.05		0.55
78	UACC-62	melanoma	Non Mutated	wt	125.08		0.56
79	C32	melanoma	Non Mutated	wt	152.32		0.75
123 – 240	NCI-H647, SK-LU-1, NCI-H1650, NCI-H2228, LU99, ...	NSCLC		p.?, p.H193R, p.?, p.Q331*, p.C176Y, ...			

Figure 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 experiment, the analyst adds the *copy number value* of *CDKN2A* as an additional column (shown in pink). 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.

replaced the LineUp technique as a core component in the Ordino drug discovery tool,[61] which is in productive use at Boehringer Ingelheim, demonstrates that the domain experts are convinced of its effectiveness and added value.

In additional high-level feedback, the domain experts mentioned that they would like to confirm the statistical significance of visual patterns they see in the overview as well as between groups of items. However, this approach could easily lead to incorrect inferences, unless some precautions are taken.[64,65] In a follow-up project, we are working on a solution that supports such confirmatory analysis in a way that users can understand without being trained in statistics.[66]

Discussion and limitations

Revisiting our discussion of visualization techniques for tabular data (overview, projection, tabular, and MCV 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 datasets 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). We also designed Taggle so that it can be used within an MCV framework.

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

Scalability

Taggle scales to more than 1 million rows on a modern browser, as demonstrated when loading the *1M Random Dataset* in the demo application. We achieve this performance by leveraging rendering optimizations, which ensure that only visible rows are processed. Although the rendering time stays almost constant, larger datasets require more time for data operations, such as sorting, grouping, or computing histograms—which always need to be done for the full dataset. The performance depends on the number of CPU cores available on the client machine, as the workload is distributed between multiple parallel web workers, if possible.

To demonstrate the computational scalability, we executed performance measurements for common operations on five datasets consisting of 100, 1000, 10,000, 100,000, and 1,000,000 data items. Each dataset consisted of one textual, two numerical, and two categorical attributes generated with uniform distribution. For each tested operation, we measured the time between triggering the operation (e.g. pressing the sort button) and the end result appearing on screen. Animation is not useful when rearranging large datasets; hence, it is disabled by default for such datasets. To make the results comparable across all datasets, we disabled animations for all conditions when benchmarking. For measurements, we used the performance profiler from Google Chrome DevTools (v. 71.0.3578.98). We repeated each measurement five times. Table 1 presents the average times in milliseconds. The tests were done on a machine with Intel

Table 1. Completion time in milliseconds for various operations using five datasets with 100 to 1 million items.

		100	1000	10,000	100,000	1,000,000
Load		529	545	611	1012	4107
Sort numerical	DM	321	338	358	643	2642
	OM	288	681	741	970	3626
Sort grouped	DM	324	324	381	518	1911
	OM	306	661	743	923	2830
Sort textual	DM	300	367	397	728	3639
	OM	302	647	730	1069	5075
Filter numerical	DM	407	415	460	598	1442
	OM	419	1883	1745	1876	2858
Filter categorical	DM	357	435	475	562	1372
	OM	403	1982	1079	1196	2261

DM indicates operations performed in detailed mode; OM indicates operations performed in overview mode.

Core i7-5930K processor (3.5 GHz, six cores), 32 GB RAM, NVIDIA GeForce GTX 970 graphics card. Note that the browser-based tracking tool may slightly decrease the actual performance.

Since the full dataset needs to be loaded into memory first, the size of the data table determines the loading time. Naturally, the number of rendered items also influences the run-time performance. Although we optimize the rendering to process only visible rows, there can be notable performance differences between detail and overview mode, since the number of rendered items is much larger in overview mode. For example, in our full-HD setup with viewport size 1387 × 882 pixel, detail mode allowed for 39 item rows, but in overview mode, we had up to 775 rows on screen. Note that due to the design decision that every item is at least one pixel high, the table grows out of the visible screen space for larger tables. Table 1 shows that for the smallest dataset, where the number of rendered elements is low in either case, the performance difference between overview and detail mode is minimal. For other datasets, the time necessary for preparing and rendering the elements is much more apparent.

Aggregation of categorical and textual attributes

Although there are numerous possibilities for aggregation for numerical data—ranging from aggregation in data space (mean, median) to spatial aggregation (box plots, histograms)—the options are limited for categorical and textual data. In our prototype, we offer three possibilities for aggregation of categorical data: a matrix, a histogram, and a distribution bar. Due to spatial restraints and the limited scalability of the color channel, there are limits with respect to the number of categories that can be sensibly encoded this way. Taggle uses a predefined color scheme with 22 distinct colors. However, if a categorical attribute has more than 22 categories, we treat the column as textual. Colors are automatically assigned, but users also have the possibility to adjust colors manually to resolve cases where colors are repeated between columns.

Aggregation of textual attributes, however, is even more limited. In our prototype implementation, we list a sample of items from the aggregated group to summarize the group's content using the order of the items. An alternative approach would be to select samples based on other criteria such as the frequency of occurrence. This approach, however, is practical only for data attributes with repetitive values.

Automatic aggregation

In the design process, we investigated methods for automatically aggregating rows and columns, with the goal of increasing scalability. For example, when in overview mode, we tried to automatically aggregate groups to make space for user-selected rows that are shown with increased height. We found, however, that users had difficulties understanding the unexpected changes and subsequently interpreting the individual items and aggregated groups. As this violated the *discriminability* design guideline proposed by Elmqvist and Fekete,[48] we removed the automatic aggregation feature. Instead, as part of future work, we plan to implement and evaluate a recommendation approach that suggests possible layout changes without automatically applying them.

Stacking of matrices and vectors

Our current prototype supports grouping of matrix columns based on a categorical attribute (see Figure 2), 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 shown, for instance, in Figures 4 and 6 and presented in Cherniack et al.[67] However, we plan to address these technical limitation in future versions.

Conclusion and future work

In this work, we presented Taggle, an item-centric, spreadsheet-like visualization technique for exploring and presenting large and complex tables. Taggle is unique among tabular data visualization techniques due to its ability to dynamically aggregate subsets of a table, which allows users to flexibly drill down into details of large tables while keeping the overview as context.

The open-source implementation presented as part of this work goes beyond a research prototype, providing a rich set of visual encodings and rendering optimizations that make it scale to a million items. Taggle can be used as a standalone tool but also integrated as a widget into MCV systems or notebook-style environments such as R Markdown or Jupyter Notebooks.

As part of future work, we plan to integrate data-driven guidance capabilities into Taggle, as implemented in StratomeX.[68] Following the idea of guided visual exploration, we plan to assist users in finding correlated attributes or similar groups based on their input.

Acknowledgements

The authors thank Bikram Kawan and Martin Ennemoser for their contributions to the initial prototype implementation as well as Christian Haslinger and Andreas Wernitznig for providing valuable conceptual feedback.

Conflict of interest

The author(s) declared the following potential conflicts of interest with respect to the research, authorship, and/or publication of this article: Samuel Gratzl, Holger Stitz, Alexander Lex, and Marc Streit hold shares and/or are employed by datavisyn GmbH, which provides its customers with support for using and deploying the open-source Taggle software.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported in part by Boehringer Ingelheim Regional Center Vienna, the State of Upper Austria (FFG #851460), the Austrian Science Fund (FWF P27975-NBL), and the National Science Foundation (NSF IIS 1751238).

ORCID iD

Katarina Furmanova https://orcid.org/0000-0003-2805-8784

Supplemental material

Supplemental material for this article is available online.

References

1. Rao R and Card SK. The table lens: merging graphical and symbolic representations in an interactive focus + context visualization for tabular information. In: *Proceedings of the SIGCHI conference on human factors in computing systems (CHI '94)*, pp. 318–322. ACM.
2. Perin C, Dragicevic P and Fekete J. Revisiting Bertin matrices: new interactions for crafting tabular visualizations. *IEEE Trans Vis Comput Graph* 2014; 20(12): 2082–2091.
3. Gratzl S, Lex A, Gehlenborg N, et al. LineUp: visual analysis of multi-attribute rankings. *IEEE Trans Vis Comput Graph* 2013; 19(12): 2277–2286.
4. Gu Z, Eils R and Schlesner M. Complex heatmaps reveal patterns and correlations in multidimensional genomic data. *Bioinformatics* 2016; 32(18): 2847–2849.
5. Munzner T. *Visualization analysis and design*. Boca Raton, FL: CRC Press; Taylor & Francis, 2014, p. 334.
6. Lex A, Schulz HJ, Streit M, et al. VisBricks: multiform visualization of large, inhomogeneous data. *IEEE Trans Vis Comput Graph* 2011; 17(12): 2291–2300.
7. Eisen MB, Spellman PT, Brown PO, et al. Cluster analysis and display of genome-wide expression patterns. *Proc Natl Acad Sci USA* 1998; 95(25): 14863–14868.
8. Dimara E, Bezerianos A and Dragicevic P. Conceptual and methodological issues in evaluating multidimensional visualizations for decision support. *IEEE Trans Vis Comput Graph* 2018; 24(1): 749–759.
9. Van der Maaten L and Hinton G. Visualizing data using t-SNE. *J Mach Learn Res* 2008; 9: 2579–2605.
10. Chuang J, Ramage D, Manning C, et al. Interpretation and trust: designing model-driven visualizations for text analysis. In: *Proceedings of the SIGCHI conference on human factors in computing systems (CHI '12)*, pp. 443–452. ACM.
11. Stahnke J, Dörk M, Müller B, et al. Probing projections: interaction techniques for interpreting arrangements and errors of dimensionality reductions. *IEEE Trans Vis Comput Graph* 2016; 22(1): 629–638.
12. Sacha D, Zhang L, Sedlmair M, et al. Visual interaction with dimensionality reduction: a structured literature analysis. *IEEE Trans Vis Comput Graph* 2017; 23(1): 241–250.
13. Wattenberg M, Viégas F and Johnson I. How to use t-SNE effectively. *Distill* 2016; 1(10): e2.
14. Liu Z, Navathe SB and Stasko JT. Ploceus: modeling, visualizing, and analyzing tabular data as networks. *Inform Visual* 2014; 13(1): 59–89.
15. Heer J and Perer A. Orion: a system for modeling, transformation and visualization of multidimensional heterogeneous networks. *Inform Visual* 2014; 13(2): 111–133.
16. Bigelow A, Nobre C, Meyer M, et al. Origraph: interactive network wrangling. In: *Proceedings of IEEE conference on visual analytics science and technology (VAST '19)*. IEEE.
17. Becker RA and Cleveland WS. Brushing scatterplots. *Technometrics* 1987; 29(2): 127–142.
18. Elmqvist N, Dragicevic P and Fekete JD. Rolling the dice: multidimensional visual exploration using scatterplot matrix navigation. *IEEE Trans Vis Comput Graph* 2008; 14(6): 1539–1148.
19. Inselberg A. *Parallel coordinates: visual multidimensional geometry and its applications*. Berlin: Springer, 2009.
20. Wegman EJ. Hyperdimensional data analysis using parallel coordinates. *J Am Stat Assoc* 1990; 85(411): 664–675.
21. Heinrich J and Weiskopf D. State of the art of parallel coordinates. In: *Eurographics 2013—state of the art reports*. Girona: The Eurographics Association, pp. 95–116.
22. Kandogan E. Visualizing multi-dimensional clusters, trends, and outliers using star coordinates. In: *Proceedings of the SIGKDD conference on knowledge discovery and data mining (SIGKDD '01)*, pp. 107–116. ACM.
23. Tominski C, Abello J and Schumann H. Axes-based visualizations with radial layouts. In: *Proceedings of the*

ACM symposium on applied computing (SAC '04), pp. 1242–1247. ACM.

24. Claessen JH and van Wijk JJ. Flexible linked axes for multivariate data visualization. *IEEE Trans Vis Comput Graph* 2011; 17(12): 2310–2316.
25. Im JF, McGuffin M and Leung R. GPLOM: the generalized plot matrix for visualizing multidimensional multivariate data. *IEEE Trans Vis Comput Graph* 2013; 19(12): 2606–2614.
26. Wilkinson L and Friendly M. The history of the cluster heat map. *Am Stat* 2009; 63(2): 179–184.
27. Liiv I. Seriation and matrix reordering methods: an historical overview. *Stat Anal Data Min* 2010; 3(2): 70–91.
28. Seo J and Shneiderman B. Interactively exploring hierarchical clustering results. *Comput* 2002; 35(7): 80–86.
29. Wu HM, Tien YJ and Chen C-h. GAP: a graphical environment for matrix visualization and cluster analysis. *Comput Stat Data An* 2010; 54(3): 767–778.
30. Caraux G and Pinloche S. PermutMatrix: a graphical environment to arrange gene expression profiles in optimal linear order. *Bioinformatics* 2005; 21(7): 1280–1281.
31. Fernandez NF, Gundersen GW, Rahman A, et al. Clustergrammer, a web-based heatmap visualization and analysis tool for high-dimensional biological data. *Sci Data* 2017; 4: 170151.
32. Blumenschein M, Behrisch M, Schmid S, et al. SMARTexplore: simplifying high-dimensional data analysis through a table-based visual analytics approach. In: *Proceedings of IEEE conference on visual analytics science and technology (VAST '18)*, pp. 36–47. IEEE.
33. Polis C. DataComb: an interface for combing through tabular datasets. *ByteMuse*, 21 June 2015, http://www.bytemuse.com/post/data-comb-visualization/
34. Goldman M. Visual spreadsheet, 2019, https://ucsc-xena.gitbook.io/project/overview-of-features/visual-spreadsheet
35. Nobre C, Gehlenborg N, Coon H, et al. Lineage: visualizing multivariate clinical data in genealogy graphs. *IEEE Trans Vis Comput Graph* 2019; 25(3): 1543–1558.
36. Nobre C, Streit M and Lex A. Juniper: a tree + table approach to multivariate graph visualization. *IEEE Trans Vis Comput Graph* 2019; 25(1): 544–554.
37. Spenke M, Beilken C and Berlage T. FOCUS: the interactive table for product comparison and selection. In: *Proceedings of the ACM symposium on user interface software and technology (UIST '96)*, pp. 41–50. ACM.
38. Spenke M and Beilken C. InfoZoom—analysing formula one racing results with an interactive data mining and visualisation tool. *WIT Trans Inform Commun Technol* 2000; 25: 455–464.
39. Weaver C. Building highly-coordinated visualizations in improvise. In: *Proceedings of the IEEE symposium on information visualization (InfoVis '04)*, pp. 159–166. IEEE.
40. Yalcin MA, Elmqvist N and Bederson BB. Keshif: rapid and expressive tabular data exploration for novices. *IEEE Trans Vis Comput Graph* 2018; 24(8): 2339–2352.
41. Baldonado MQW, Woodruff A and Kuchinsky A. Guidelines for using multiple views in information visualization. In: *Proceedings of the working conference on advanced visual interfaces (AVI '00)*, pp. 110–119. ACM.
42. Henry N, Fekete JD and McGuffin MJ. NodeTrix: a hybrid visualization of social networks. *IEEE Trans Vis Comput Graph* 2007; 13(6): 1302–1309.
43. Lex A, Streit M, Schulz HJ, et al. StratomeX: visual analysis of large-scale heterogeneous genomics data for cancer subtype characterization. *Comput Graph Forum* 2012; 31(33): 1175–1184.
44. Kern M, Lex A, Gehlenborg N, et al. Interactive visual exploration and refinement of cluster assignments. *BMC Bioinformatics* 2017; 18(1): 406.
45. Gratzl S, Gehlenborg N, Lex A, et al. Domino: extracting, comparing, and manipulating subsets across multiple tabular datasets. *IEEE Trans Vis Comput Graph* 2014; 20(12): 2023–2032.
46. Streit M, Gratzl S, Gillhofer M, et al. Furby: fuzzy force-directed bicluster visualization. *BMC Bioinformatics* 2014; 15(Suppl. 6): S4.
47. Lex A, Streit M, Partl C, et al. Comparative analysis of multidimensional, quantitative data. *IEEE Trans Vis Comput Graph* 2010; 16(6): 1027–1035.
48. Elmqvist N and Fekete JD. Hierarchical aggregation for information visualization: overview, techniques, and design guidelines. *IEEE Trans Vis Comput Graph* 2010; 16(3): 439–454.
49. Fua YH, Ward MO and Rundensteiner EA. Hierarchical parallel coordinates for exploration of large datasets. In: *Proceedings of the IEEE conference on visualization (Vis '99)*, pp. 43–50. IEEE.
50. Heinrich J, Vehlow C, Battke F, et al. iHAT: interactive hierarchical aggregation table for genetic association data. *BMC Bioinformatics* 2012; 13(Suppl. 8): S2.
51. Holzhüter C, Lex A, Schmalstieg D, et al. Visualizing uncertainty in biological expression data. In: *Proceedings of the SPIE conference on visualization and data analysis, IS&T/SPIE*, p. 82940O.
52. Conklin N, Prabhakar S and North C. Multiple foci drill-down through tuple and attribute aggregation polyarchies in tabular data. In: *Proceedings of the IEEE symposium on information visualization (InfoVis '02)*, pp. 131–134. IEEE.
53. Lex A, Gehlenborg N, Strobelt H, et al. UpSet: visualization of intersecting sets. *IEEE Trans Vis Comput Graph* 2014; 20(12): 1983–1992.
54. Furnas GW. Generalized fisheye views. In: *Proceedings of the ACM SIGCHI conference on human factors in computing systems (CHI '86)*, pp. 16–23. ACM.
55. Tyner C, Barber GP, Casper J, et al. The UCSC genome browser database: 2017 update. *Nucleic Acids Res* 2017; 45(D1): D626–D634.
56. Streit M and Gehlenborg N. Points of view: bar charts and box plots. *Nat Methods* 2014; 11(2): 117.
57. Heer J and Robertson GG. Animated transitions in statistical data graphics. *IEEE Trans Vis Comput Graph* 2007; 13(6): 1240–1247.
58. Carenini G and Lloyd J. ValueCharts: analyzing linear models expressing preferences and evaluations. In:

Proceedings of the conference on advanced visual interfaces (AVI '04), pp. 150–157. ACM.
59. Tufte E. *The visual display of quantitative information*. 2nd ed. Cheshire, CT: Graphics Press, 2001, p. 156.
60. Kosara R, Bendix F and Hauser H. Parallel sets: interactive exploration and visual analysis of categorical data. *IEEE Trans Vis Comput Graph* 2006; 12(4): 558–568.
61. Streit M, Gratzl S, Stitz H, et al. Ordino: a visual cancer analysis tool for ranking and exploring genes, cell lines and tissue samples. *Bioinformatics* 2019; 35: 3140–3142.
62. Jeay S, Gaulis S, Ferretti S, et al. A distinct p53 target gene set predicts for response to the selective p53-HDM2 inhibitor NVP-CGM097. *Elife* 2015; 4: e06498.
63. Barretina J, Caponigro G, Stransky N, et al. The cancer cell line encyclopedia enables predictive modelling of anticancer drug sensitivity. *Nature* 2012; 483(7391): 603–607.
64. Dwork C, Feldman V, Hardt M, et al. The reusable holdout: preserving validity in adaptive data analysis. *Science* 2015; 349(6248): 636–638.
65. Zgraggen E, Zhao Z, Zeleznik R, et al. Investigating the effect of the multiple comparisons problem in visual analysis. In: *Proceedings of the SIGCHI conference on human factors in computing systems (CHI '18)*, p. 479. ACM.
66. Eckelt K, Adelberger P, Zichner T, et al. TourDino: a support view for confirming patterns in tabular data. In: *Proceedings of the EuroVis workshop on visual analytics (EuroVA' 19)*, pp. 7–11.
67. Cherniack AD, Shen H, Walter V, et al. Integrated molecular characterization of uterine carcinosarcoma. *Cancer Cell* 2017; 31(3): 411–423.
68. Streit M, Lex A, Gratzl S, et al. Guided visual exploration of genomic stratifications in cancer. *Nat Methods* 2014; 11(9): 884–885.