Blaeu: Mapping and Navigating Large Tables with Cluster Analysis

Thibault Sellam
CWI
the Netherlands
thibault.sellam@cwi.nl

Robin Cijvat
MonetDB Solutions
the Netherlands
robin.cijvat@monetdbsolutions.nl

Richard Koopmanschap
MonetDB Solutions
the Netherlands
koopmans@monetdbsolutions.nl

Martin Kersten
CWI
the Netherlands
martin.kersten@cwi.nl

ABSTRACT

Blaeu is an interactive database exploration tool. Its aim is to guide casual users through large data tables, ultimately triggering insights and serendipity. To do so, it relies on a double cluster analysis mechanism. It clusters the data vertically: it detects *themes*, groups of mutually dependent columns that highlight one aspect of the data. Then it clusters the data horizontally. For each theme, it produces a *data map*, an interactive visualization of the clusters in the table. The data maps summarize the data. They provide a visual synopsis of the clusters, as well as facilities to inspect their content and annotate them. But they also let the users navigate further. Our explorers can change the active set of columns or drill down into the clusters to refine their selection. Our prototype is fully operational, ready to deliver insights from complex databases.

1. INTRODUCTION

Data explorers query a database to discover its content. For these users, the holy grail is the *nugget*, a combination of query and visualization that highlights a phenomenon that they neither knew, nor expected. Exploration is called for when users discover a new source of data. Then, they must start from scratch: which tuples and columns should they inspect? And how should they visualize them? Our explorers may not even know where to begin. Typically, they must "play" with the data, or they must "tinker" with it to get a mental picture of what it contains [1]. Database exploration also comes into the picture when users already have a specific task in mind, but they do not know how to cast it into a precise query. For instance, suppose that we have access to a database of countries, containing a few dozen economic, social and well-being indicators. We want to find the countries with the best work conditions. How can we describe

such a vague requirement with a precise SQL query? This task is difficult because it requires background knowledge. Our users need to know which variables to use: shall they use the number of hours worked, the average salary, or the general satisfaction of the employees? Also, which threshold shall they set, i.e., how much satisfaction is good enough? In fact, formulating the question with SQL is a significant problem in itself.

The straightforward approach to database exploration is trial and error. Users write a query, study the result, and modify the query subsequently. They carry out this process in SQL directly, or through graphical tools such as Tableau [5]. This approach is efficient for small datasets, but what if the selection contains 100s columns and 100,000s tuples? Such large result sets are extremely difficult to visualize. Large tables are overwhelming, and no classic multivariate visualization method (e.g., scatter-plot matrices, parallel coordinates) can effectively deal with this volume. Furthermore, the space of all possible query refinements is huge, and therefore manual effort can quickly become tedious and inefficient. How can we provide *guidance* to our data explorers?

In this demo proposal, we present Blaeu, our database browser. The main idea is to guide the users through *cluster analysis* (or *clustering*), a data mining technique that partitions a dataset such that similar objects are grouped. For a given query, Blaeu exploits clustering in two ways. First, it clusters the data vertically, to help users choose variables. For a given query, Blaeu detects *themes*, that is, groups of columns which describe the same "aspect" of the data. In our example, a theme could describe "economic indicators", "labor statistics", or "well-being indices". Next, it clusters the data horizontally, to help users refine their selection. For each theme, Blaeu generates a *data map*, that is, an interactive visualization of the clusters in the query results. This map summarizes the data, and it presents alternatives for further refinements. For a smooth exploration experience, Blaeu must cluster millions of tuples on hundreds of columns at interaction time. To do so, it relies on multi-scale sampling and low-level data sharing techniques, supported by the high performance DBMS MonetDB and its connector with the statistical package R.

We introduced the concepts of maps and themes in a previous publication [4]. During this demo, we will demonstrate

(a) List of Themes.

(b) Data map associated with the first theme.

(c) Map after a zoom and a highlight.

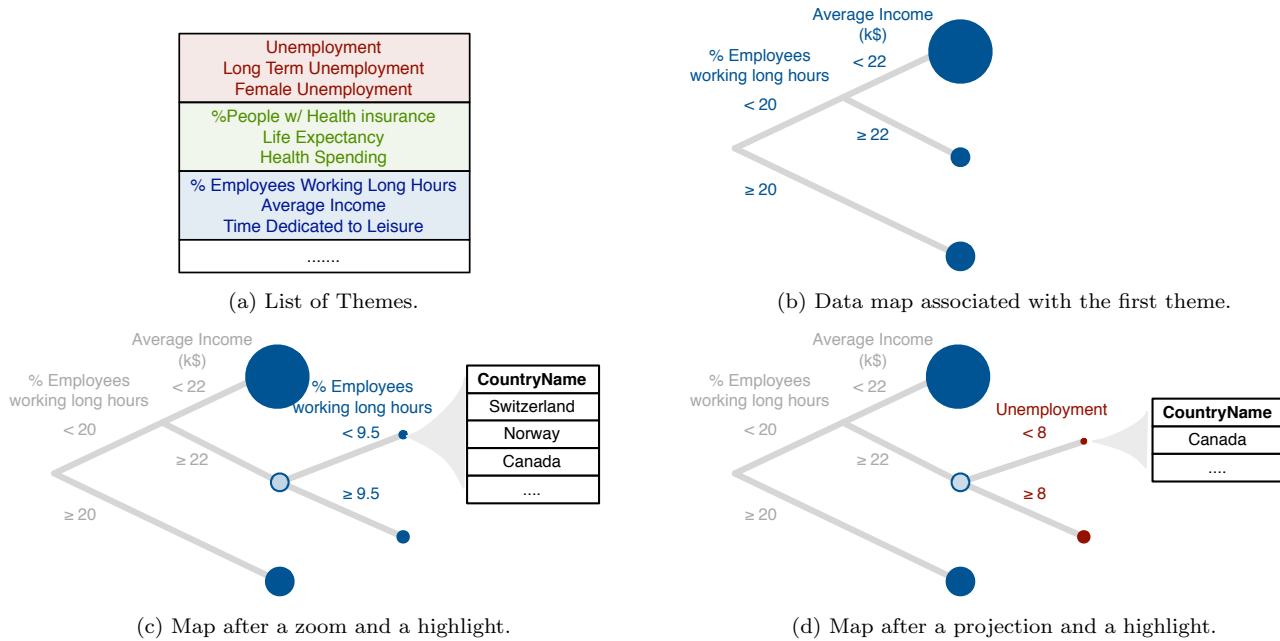(d) Map after a projection and a highlight.

Figure 1: Writing Queries With Blaeu.

them with real-life data sets of various sizes and complexities. We will confront our visitors with unknown datasets, and encourage them to navigate the clusters derived by Blaeu. We will show that Blaeu allows fast, keyboard-free exploration with little to no prior knowledge of the data.

2. THE DATA MAP MODEL

The key concepts behind Blaeu are themes and data maps. Let us present them in more detail, using the aforementioned example. Our users have access to a database of countries. They are seeking the countries with the best working conditions, but they do not precisely know which columns and tuples this task involves. How can we help them?

Figure 1a presents a few of Blaeu's themes. Each theme is a vertical slice of the database, giving a particular "view" over the tuples. The first theme refers to unemployment statistics. The second theme describes health indicators. The third one contains columns related to labor conditions: this is the theme that interests our users. Accordingly, they select it.

Once the users have selected a theme, Blaeu detects clusters in the underlying data, and it returns a map. Figure 1b presents the map relevant to our example. Our system detected three clusters, organized in a hierarchy. At the highest level, it identifies two regions: countries with many employees working long hours (more than 20% of all employees), and other countries. It subdivides the latter group in two smaller groups: countries with high average incomes, and countries with low average incomes. The area of the leaves shows the number of tuples covered.

To interact with the data map, users can perform four navigational actions: highlight, zoom, project or rollback.

- The *zoom* lets users "drill down" in a cluster, as shown in Figure 1c. In this example, the users zoomed into the middle region, highlighted in light blue (low working hours, high wages). Blaeu subdivides it in two clusters.

- The *highlight* operation allows users to inspect the tuples inside each region. In Figure 1c, they highlight the name of the countries: we see that Switzerland, Canada and Norway appear as countries with high incomes and relatively low working hours. For more details, our prototype provides classic univariate and bivariate visualization methods, such as histograms and scatter-plots.

- With the *projection*, the users can refine their map with the columns of another theme. In Figure 1d, they select unemployment indicators, which highlight an alternative aspect of the data.

- Each action is reversible, and the users can always go back to a previous state of the system with a *rollback*.

Observe that data maps serve both as output and input. They serve as output because they summarize the data: instead of showing raw tuples, Blaeu displays high-level groups of similar tuples. They serve as input because they are interactive: our users can click on the regions to refine their queries, or modify the active selection of columns with projections and highlights.

With Blaeu, our users implicitly formulate and refine Select-Project queries. The range of queries that they can express is smaller than that directly offered by SQL - they are limited to the clusters in the data. But this limitation has practical advantages: the users can study each cluster in turn, and chose the one that seems the most promising. In essence, Blaeu *quantizes* the query space: to refine their queries, the users need only to consider a few discrete alternatives, instead of a wide, continuous space. We refer the interested reader to our journal article for a full study of Blaeu's expressivity [4].
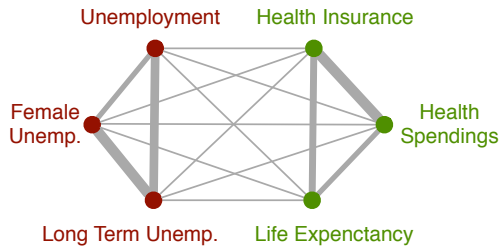
Figure 2: Example of Dependency Graph.

3. GENERATING MAPS

We presented the main concepts behind Blaeu: the themes, the data map and the navigational actions. Let us now focus on the mapping engine itself.

Creating Themes. Blaeu creates themes in two steps. First, it generates a *dependency graph*, a weighted undirected graph in which each vertex represents a column and each edge the statistical dependency between two columns. Figure 2 presents the graph associated to our running example. To measure the dependency between the columns, we could have used any function from the literature, such as the correlation coefficient or the mutual information [4]. In our implementation we chose the latter because it is very flexible: it copes with mixed values and it is sensitive to non-linear relationships.

During the second step, Blaeu creates groups of mutually dependent columns. To do so, it partitions the dependency graph with cluster analysis. To implement this step, we had to chose between a dozens clustering algorithms from the literature. We chose Partitioning Around Medoids (PAM) because it is accurate, well established and fast enough [3]. The PAM algorithm is a *k-medoid* algorithm: it creates partitions that minimize the aggregated weight between the vertices and their medoid, i.e., the most "central" element in their partition.

Creating Maps. In theory, Blaeu could build data maps with any clustering algorithm. However, it must consider three practical requirements. First, it must cope with mixed data, potentially including missing values. Second, it must be able to detect arbitrarily shaped clusters. Finally, its results should be easy to describe and to understand. This excludes complex shapes in multidimensional spaces. Observe that the last two requirements are somewhat contradictory: how can we detect complex clusters but output them in a simple form?

Our idea is to pipeline three stages: preprocessing, cluster detection and cluster description. Figure 3 illustrates this process. Preprocessing is necessary to clean the data and tackle mixed types. Blaeu removes the primary keys, it normalizes the continuous variables, and it introduces dummy binary variables to represent the categorical data (each dummy variable corresponds to one category). The result of this operation is a set of vectors, where each vector represents a tuple in the database. During the second step, Blaeu clusters these vectors. As previously, it uses the PAM algorithm [3, 4]. We obtain one cluster ID for each tuple. During the final step, Blaeu simplifies the clusters. To do so, it uses a decision tree algorithm, such as CART [2]. It trains the tree model on the original tuples from the database, using the cluster IDs obtained previously as class labels.
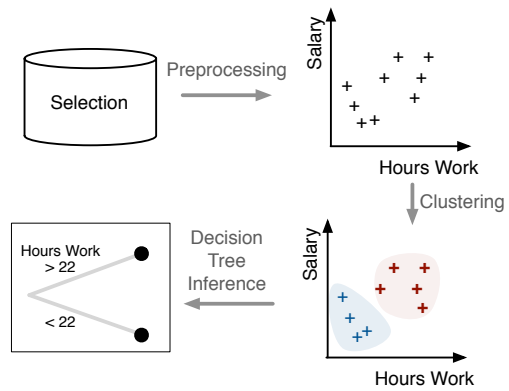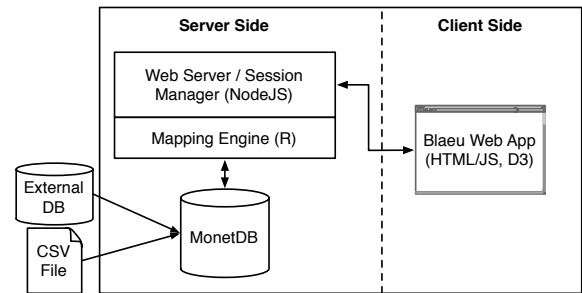

Figure 3: Blaeu's mapping algorithm.

Figure 4: Architecture of Blaeu's implementation.

The advantage of our approach is that it gives the best of both clustering and decision trees. On one hand, we can use arbitrarily sophisticated cluster detection algorithms. On the other hand, Blaeu's results are always interpretable. The downside of our approach is that it induces a loss of accuracy: the decision tree only approximates the real partitions detected during the clustering step.

Number of clusters. When creating themes and maps, we use partitional clustering algorithms, that is, algorithms which divide the data in k groups. Nevertheless, detecting the "right" number of clusters k is not straightforward. In our implementation, we use the *silhouette coefficient* from the statistics literature [3]. The silhouette coefficient describes the quality of a cluster: it defines how well each data point fits within its partition. This information is useful in itself, because it helps users understand their results. But it also helps us choose a k: we generate several partitionings with different numbers of clusters, and keep the one with the best score. We refer the reader to the original reference for the more details about the silhouette coefficient.

Sampling. All the operations involved in Blaeu's pipeline are time consuming. To keep the latency low, our system relies heavily on sampling. After each zoom, Blaeu only takes a few thousand samples from the database. Our experiments reveal that the loss of accuracy is minimal [4]. Furthermore, our system uses probabilistic algorithms. For instance, it computes the silhouette scores in a Monte-Carlo fashion: it extracts a few sub-samples from the user's selection, computes the clustering quality of those, and averages the results. Similarly, when the data is too large, Blaeu creates the maps with CLARA [3], a sampling-based variant of the PAM algorithm.
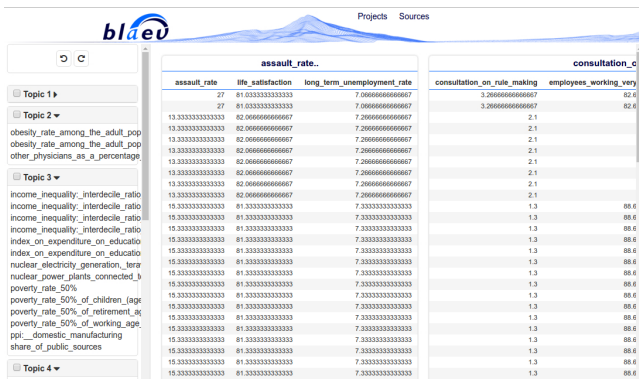
Figure 5: Blaeu's theme view. The left panel lists the columns in each theme. When the users select a theme, the corresponding view of the database appears in the central panel.

Figure 6: Blaeu's map view. The main panel displays the map. The left panel displays information about the active region.

4. BLAEU DEMONSTRATION

We now present the architecture of our implementation, its interface and how we intend to demonstrate it.

4.1 Architecture of the System

We implemented Blaeu as a Web application, comprising a DBMS, a two-tiered server and browser code as shown in Figure 4. The database is MonetDB, used for the storage and the sampling of the user's data. The bottom layer of the server collects data from the DBMS and produces the maps. We implemented it in R. The top layer of the server manages the sessions and relays the maps to the clients. It uses NodeJS. On the client side, users interacts with a HTML/JavaScript application, which produces the charts and transmits the user's actions. Our code relies heavily on D3, a library to produce interactive visualizations.

Figures 5 and 6 present two screenshots of the system. The first screenshot depicts Blaeu's *theme view*, through which users can browse and edit the themes. The second screenshot presents Blaeu's *map view*, which displays the actual data map.

4.2 Demonstration Scenarios

We will successively demonstrate Blaeu with three databases, presented in increasing order of size and complexity.

- The **Hollywood** dataset presents data about 900 Hollywood movies released between 2007 and 2013. It contains 12 columns. Which films are the most profitable? Which are those that fail? How do critics and commercial success relate to each other? Through this simple database, our visitors will discover Blaeu's concepts and build elementary queries.
- The **Countries and Work** use case is based on public data sets from the OECD, an international organization[1]. It describes economic performance indicators, labor statistics and well-being indices for more than 1,500 regions belonging to 31 different countries. It contains 6,823 rows and 378 columns. Thanks to this dataset, we will demonstrate how to build complex queries. Among others, our users will discover why working in Canada is generally a good idea, and they
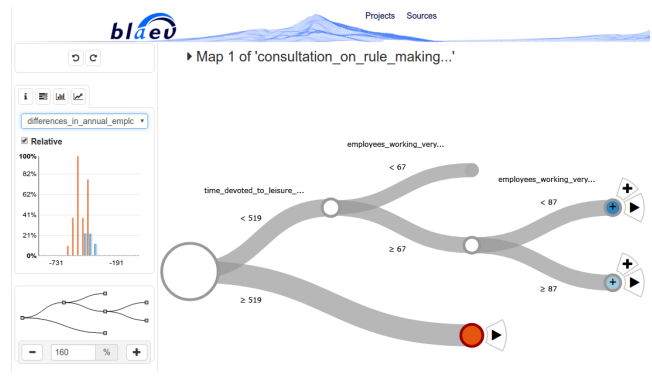
_____

[1] http://www.oecd.org/

will compare the work life in Amsterdam with that in the Bay Area.

- The **LOFAR** database is the result of a large-scale radio astronomy experiment in the Netherlands. It describes the positional and physical properties of light sources (e.g., stars). The exact dimensions of the dataset are still unknown, but we expect it to contain 100,000s of tuples and several dozens variables. Through this use case, our visitors will experience Blaeu with a large, complex dataset.

Throughout the demonstration, we will encourage the participants to interact with Blaeu themselves.

5. CONCLUSION

In this demo paper, we presented Blaeu, an innovative data exploration system based on multi-scale cluster analysis. We presented the main concepts behind our system: the themes, the maps and their associated actions. We look forward to let visitors interact with our prototype.

6. ACKNOWLEDGMENTS

7. REFERENCES

[1] A. Abouzied, J. Hellerstein, and A. Silberschatz. Dataplay: interactive tweaking and example-driven correction of graphical database queries. In *Proc. UIST*, pages 207–218. ACM, 2012.

[2] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and regression trees*. CRC press, 1984.

[3] L. Kaufman and P. J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 1990.

[4] T. Sellam and M. Kersten. Cluster-driven navigation of the query space. *IEEE TKDE*, 28(5):1118 – 1131, 2016.

[5] C. Stolte, D. Tang, and P. Hanrahan. Polaris: a system for query, analysis, and visualization of multidimensional relational databases. *TVCG*, 2002.