From Multiple Linked Views to Multiple Linked Analyses: the Meme Media Digital Dashboard

Jonas Sjöbergh and Yuzuru Tanaka
Meme Media Lab
Hokkaido University
Sapporo, Japan
Email: {js, tanaka}@meme.hokudai.ac.jp

***Abstract*—We describe a system called the *Digital Dashboard* that uses multiple linked views of data. All views allow interaction with the visualization results and interaction is done through direct manipulation. The system has been extended to allow new complex data to be generated in analysis components at runtime, e.g. by statistical analysis or data mining of parts of the data. The resulting data can be used in other linked views or analysis components, so when e.g. a data mining parameter is changed, all linked views (or analysis components) are automatically updated as soon as the new calculations are finished, and when something changes in linked components (e.g. a different subset of the data is selected), the calculations are automatically redone (if necessary).**

***Keywords*-visualization; visual analytics; explorative data visualization; multiple linked views; direct manipulation;**

I. Introduction

Recently, we have encountered similar problems in two quite different projects. One project concerns “personalized medicine”, trying to develop methods to give cancer patients treatments tailored specifically to them instead of giving all patients more or less the same treatment. The other project concerns snow removal in the city of Sapporo. In both cases, we have problems with sparse, low quality, and very high dimensional data. The problems are also not well understood and hence difficult to model. We believe that for hypothesis generation and for gaining insights into how to model (parts of) the problems, visual exploration of the data can be helpful.

Multiple linked views[1] of the same data is useful when visually exploring data. Interaction in one view, e.g. selecting or grouping the data, is automatically reflected in other linked views. We have previously developed a system[2] where you can use direct manipulation [1], [3] to interact with visualization results. The system supports multiple linked views and also allows data mash-up, i.e. you can visualize and link data from different sources.

Directly interacting with the visualization results that seem to tell you something interesting is very powerful. In our system, all components allow interaction and the interactions are reflected in all linked components. New visualization components and data sources can be added at any time. We believe that allowing not only interaction with linked visualization results, but also allowing linked components that can do advanced analysis can be even more powerful. We call this Exploratory Visual Analytics.

Our system has been extended to allow linking of both views and analysis components. Components can generate new complex data at runtime, and this new data can be immediately used. One example is selecting a subset of the data using a visualization component, performing machine learning on this subset, and visualizing the results in yet another component. The visualized machine learning results can be interacted with and data can be grouped based on the results etc. Analysis components are linked in the same way as other components, so changing some machine learning parameter will result in all linked views being updated with the new results as soon as the calculations are finished, and interacting with linked components that the machine learning calculations are based on will automatically trigger recalculations in the machine learning component.

II. Exploratory Visual Analytics

Exploratory Visual Analytics uses the coordinated multiple view visualization framework (multiple linked views)[1] as a base. The framework allows more than one view of the same database. In Figure 1, four different views, V_1 through V_4 based on the database queries Q_1 through Q_4, of the same database Δ are shown. This schema corresponds to the concrete visualization shown in Figure 2.

A view V_i can be a map view, a graph representation, a calendar view, etc. and shows the result $Q_i(\Delta)$ of some query Q_i associated with this view. Each view allows users to select a set of visualized objects by directly specifying each of them or by enclosing groups of objects. This defines a new additional quantification condition C to quantify the objects stored in the underlying database Δ, and this quantification defines a new database view Δ', where $\Delta' =$ select * from Δ where C.

All other linked views V_j then immediately change their visualizations from $Q_j(\Delta)$ to $Q_j(\Delta')$, or simply highlight the objects in $Q_j(\Delta')$, depending on the user specified visualization mode of V_j.

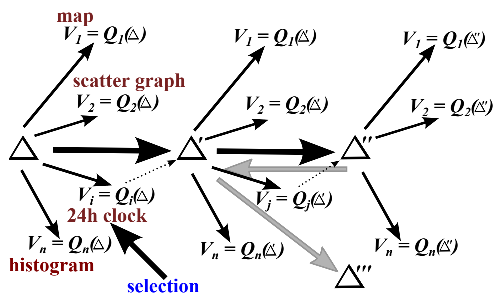


Figure 1. Multiple linked views and iterative visual exploration.

A user may start with the original database Δ and repetitively try different selections or groupings of visual objects using different views to explore the data and try to find meaningful groups of database objects. The user may roll back previous object selections to try a different quantification using the same or a different view. Figure 1 schematically shows this process.

A view in a coordinated multiple view visualization is normally just a database visualization. We would like to allow analysis tools to be applied to such quantified sets of database objects as described above, and the results of the analyses to be used in further visualizations or analyses.

So an exploratory visual analytics system requires both a coordinated multiple view visualization framework to integrate the analysis tools in, and a library of analysis tools. This allows the coordination of visualization results, quantifications based on the visualizations, analysis tools, and new data generated from these analyses. We have extended our system, described later, to allow this.

III. Comparisons to Similar Systems

There are many systems for visualization and exploration of data that use linked views[1]. Our system has the following important features:

- it uses multiple linked views of the data
- it allows interaction with all visualization results through direct manipulation
- it allows creation of new complex data at runtime, and such data to be used like any other data
- it immediately updates all linked components, allowing real time interaction and exploration
- it allows addition of new data components (data mash-up) and new visualization components at any time.

The system also has the following features that are useful, though not directly related to data exploration:

- it is component based with components that are hidden from each other by a common interface
- it allows external software to be used as components
- it runs in a Web browser (no installation necessary)
- it allows fast prototyping since it is built using a pluggable component framework called *Webbles*[4], the latest version of *Meme Media*[5].

There are systems that have several of the features above, many systems for example support data mash-up, but we are not aware of any systems that have all these features. We believe that allowing advanced analysis components to create new complex data at runtime, and such components to be linked like any other component, is a useful extension of linked views. We have not seen any systems that allow interaction with visualization results that then affect the input to such analysis components, and that also allow further visualization of and interaction with the results, and where changes in any linked components affect all other linked components (not only one-way flows).

There are systems that use graphical interfaces such as a flow chart style interface to set up what types of preprocessing, analysis, data mining, and visualization etc. should be done. Changes in the flow chart lead to updated visualizations, but interaction with the actual visualization results is very limited or not possible. Examples of such systems include *RapidMiner*[6] and *DEVise*[7].

There are systems that use multiple linked views of data and allow interaction with the visualization results. Selecting data in one view updates the other views to show only this selection, or clicking on one item in one view shows details about it in another view, etc. Creating new complex data in analysis components is not possible, though. Examples of such systems include: *SpotFire*[8], *Tioga-2*[9] (now *Tioga DataSplash*), and *Snap-Together Visualization*[10].

One example system quite similar to ours is *KNIME*[11], which uses a graphical flow chart to set up processing and visualization of data. It allows adding new user built components and uses multiple linked views for visualization. Selecting subsets of data in one view highlights these in other views, but unlike in our system it does not trigger recalculation of related data mining results etc.

Another system with many of the features above is *Orange*[12]. It sets up data flows in a flow chart, has both visualization and analysis components, allows user built components, and selections in a visualization component can trigger recalculation in data mining components etc. Unlike in our system, two components cannot feed back into each other, so selections in one component can affect the other, but selections in the other component cannot be reflected back to the first.

There are also some precursors to our system. The *VERD*[13] system is based on *IntelligentBox*, another version of *Meme Media*[5]. It visualizes relational databases and allows Web resources to be treated as relational schema. Unlike our system, visualizations are set up in a flow and interaction with visualization results only affect other visualizations "downstream" of the interaction point.

The *Trial Outline Builder*[14] is a support system for clinical trials built using *Webbles*[4] that has a data analysis part similar to our system. That system and previous versions of our system[2] used only visualization components.

IV. The Digital Dashboard System

We here describe a system called the *Digital Dashboard*. It is a tool set for data analysis, data visualization, and data exploration. The *Digital Dashboard* is designed for ad-hoc federation of components, both data components, analysis components, and visualization components. New data can be added at any time, e.g. to complement the data currently explored with more information. New analysis tools or visualization components can also be added, to add new ways of looking at the same data or to view different subsets of the data side by side.

Analysis tools can be used to create new (complex) features at runtime, and these can immediately be used in other components. All views and analysis components can be linked. Changes, e.g. subsets of data being selected or machine learning parameters being changed, are automatically reflected in all linked components immediately, allowing interactive exploration of the data. All components also allow interaction, which is done through direct manipulation.

Data components can be files, e.g. XML files or CSV files with data, or wrap a database or a Web service.

Several demo versions of the system are available online[1]. Most of the data we have are sensitive and cannot be shared, so there is not much data to use the system with, though.

A. Underlying Technology

The *Digital Dashboard* is built using pluggable software components called *Webbles*[4]. *Webbles* are the latest version of the *Meme Media* (*IntelligentPad*) framework[5] and *Webbles* run inside a Web browser. The current version requires a Microsoft Silverlight browser plugin but other than that, any Web browser can be used. In the near future, the *Webble* framework and our system will be moved to HTML5 and Javascript, supported by more platforms than Silverlight.

Webbles are like software LEGO blocks, and you can plug any two *Webbles* together and they will start sharing data and functionality. The goal of *Meme Media* is to make sharing, reediting, and redistribution of functionality as easy as copy-pasting text or images already is.

A *Webble* has ports called *slots*. *Slots* represent internal variables or input and output functions. A *Webble* can be connected to another *Webble* by making it a child to that *Webble*. A *Webble* can have any number of children, but can have at most one parent. A child can connect one of its *slots* to one of the parent's *slots* and the *Webbles* will then communicate using these *slots*. If the *slot* represents a variable, the *Webbles* will in effect have a shared variable.

In general, one visualization or analysis component in our system corresponds to one *Webble*, but some components are built using more than one *Webble*. One strength of the *Webble* framework is that you can wrap existing (non-*Webble*) software with a *Webble* wrapper. Once software has been wrapped, it can be used like any other *Webble* and other *Webbles* can connect to it without knowing that it is actually some external software. In our system, we have wrapped the ArcGIS[2] software with a *Webble* wrapping interface, allowing it to be used like a normal component.

Using these generic components meant that it was easy to add new components once the basic system was finished, and prototyping new components is very fast. When developing new components, devising a user interface for the interaction has generally been more time consuming than implementing the visualization or analysis algorithms. For this, fast prototyping has been helpful.

B. System Structure

Apart from using the pluggable component framework described in the previous Section, the system also has a simple component interface, a set of *slots*, that it expects all plugins to follow. As long as a component follows the expected interface, it can be added to the system and used immediately, without having to rebuild or even restart the system. The system itself need not be aware of the internal workings of any new components, and all components are hidden from each other by the parent system.

This means that visualization components need not worry about the format the data may be stored in or if data comes from several different sources etc. They also do not need to care about whether there are any other components visualizing the same data or not.

It is the job of data source components to hide the underlying data format and present data in a standard format. The *Digital Dashboard* parent also does not care about the data format, and as long as a visualization component and a data source component both understand a certain data type (and can agree on the name) data types the *Digital Dashboard* has never heard of can also be used.

In the system there is a *Digital Dashboard* parent *Webble*. All data source *Webbles*, all visualization component *Webbles*, and all analysis component *Webbles* are connected as children to the *Dashboard* parent. Complex components can have other *Webbles* as children in turn, but should not have other components as children.

Data source components are expected to have these *slots*:

- ProvidedFormat, XML detailing the data this source provides: the data types of the different data fields and what the fields should be called (in menus etc.).
- FormatChanged, signaling that the format of the data this source provides has changed.
- DataValuesChanged, signaling that the data has changed and the parent needs to update any visualization components using data from this source.

They also have *slots* containing the actual data.

[1]`http://cow.meme.hokudai.ac.jp/WebbleWorldPortal/`

[2]ESRI (2012) ArcGIS API for Silverlight (ver. 2.4) `http://help.arcgis.com/en/webapi/silverlight/`

Visualization components are expected to have these *slots*:

- ExpectedFormat, XML describing what types of data the plugin wants and in which *slots* it expects to receive the data. Several different sets of data can be specified (e.g. a component could accept vector data as a start and end point pair or as a starting point, a direction, and a length).
- DataValuesSetFilled, a slot where the parent informs the plugin about which slots it has filled with data (data fields can be optional).
- DataValuesChanged, a slot where the parent signals that the data has changed.
- LocalSelections, a slot where the plugin tells the parent which data items are selected/deselected/grouped together on this component.
- GlobalSelections, a slot where the parent tells the plugin the global selection status of the data items (e.g. a data item may be selected locally but unselected on another component, making it unselected globally).
- GroupColors, information on what colors the plugins should use to visualize different groups of data for a uniform look across all components.

They also have *slots* for data input, described in the XML of the ExpectedFormat *slot*.

Analysis components are generally expected to follow both the data source interface and the visualization component interface, since they will receive data like visualization components and produce data like data source components. All components are also required to have a PluginName *slot* specifying what name to use in menus etc., and a PluginType *slot* specifying what type of plugin it is.

C. Basic Usage Example

Here we give a simple usage example, showing the basic user interaction. It shows a small subset of the *Digital Dashboard* functionality: linked views (a change in one changes all) and ad hoc addition of new data.

The example uses data from a project on snow removal we are involved in. Sapporo has 2 million citizens and gets 6 meters of snow per year. It costs around 150 million dollars per year to remove snow and keep the city running.

We begin with a set of probe car data. These are data from private cars that every few seconds report where the car is, what time it is, and how fast the car is going. The data is collected using the car navigation system and the cell phone network. For privacy reasons, the data has been averaged. In this example, the city of Sapporo was divided into square cells and the number of cars passing each cell during two hour intervals was counted. Any cell with less than 10 cars was discarded for privacy reasons.

The data we used were the average speed, the number of cars, the speed compared to the average speed in the same cell at the same time of day during the summer (i.e. how much worse the speed is compared to traffic when there is no snow), and the speed compared to the winter average speed. We also use social networking data from Twitter. The data have the times, the locations, the text contents, and the user IDs of all location tagged tweets from Sapporo.

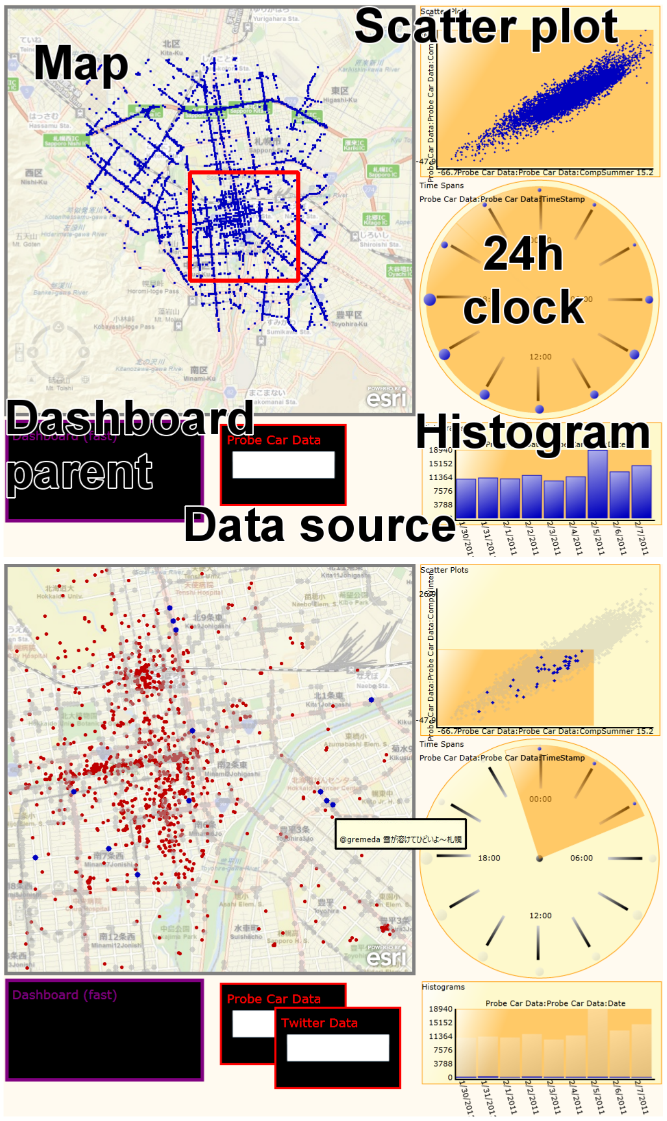


Figure 2. Basic usage with multiple linked views. Traffic data from Sapporo is explored by selecting data with speed much lower than normal (scatter plot), and then further restricting the selection to night time data (24h clock). The map is zoomed in on one problem area with many remaining blue dots (area marked with a red square). A new data source with Twitter data (red dots) is added. Mouse-over text shows people in the problem area talking about snow problems.

In Figure 2 one week of probe car data is visualized using four visualization components. There is a map with the locations, a scatter plot showing the speed difference compared to summer conditions (horizontal axis) and to average winter conditions (vertical axis), a 24 hour clock showing the amount of data at different times of day, and a histogram showing the number of data samples per day.

In this image we can see that there are few cars driving around at night (smaller dots during the night time intervals on the 24h clock) and that the traffic peak is at around 18:00. We can also see that more people use their cars on Saturdays (the big peak in the bar chart) but other than that there is not so much variation over the week.

To find locations with snow removal problems, we use the scatter plot to select only data where the speed is much lower than the normal speed. This immediately updates all visualization components. We also use the 24h clock to further restrict our selection to night time data, again updating all views. Locations with a big drop in speed even at night are likely to have problems with snow or ice. We also zoom in the map to take a closer look at an area (marked with a red square) where several locations have speed downs at night, resulting in the lower image in Figure 2.

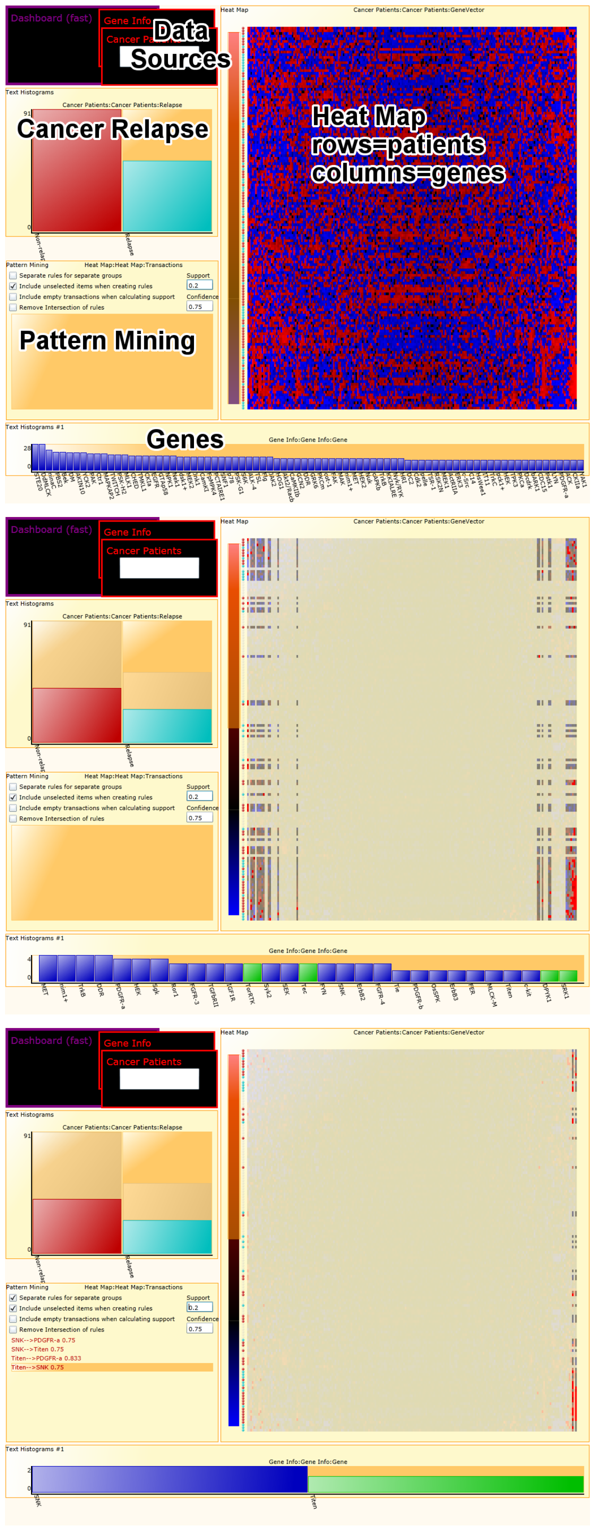
Traffic problems can be caused by other things too (e.g. construction work). To check our hypothesis that this area has problems with snow, we drag-and-drop another data source into the *Dashboard*, adding the Twitter data. The Twitter data locations are shown with red dots on the map, and holding the mouse pointer over one tweet near our problem area shows the text content: "The melting snow in Sapporo is horrible!" (translated to English). The Twitter data seem to confirm that the problems are snow related.

D. Usage Example with Analysis Components

In the next example patient data is used. For each patient there are gene response data and a field indicating if the cancer relapsed after treatment or not. In Figure 3, the gene response data is shown in a heat map, where the rows are patients and the columns are genes. Red color means a high response, black color an average response, and blue color means a very low response. The relapse or no relapse status is shown in a bar chart, and another bar chart shows individual genes and can be used to access more information about genes that are deemed interesting. We also have an analysis component, a frequent pattern mining component, linked to the other components.

The patients are grouped into patients with or without relapse, using the bar chart component. In the middle image, the heat map is used to create item sets for the frequent pattern mining. Only cells with a high response are selected by setting a response threshold. Transaction data are generated where each patient is treated as a transaction, and this transaction is made up by all the genes that for this patient had a response above the threshold. So a patient with a high response for (only) the genes "SNK", "Titen", and "TrkB" would generate a transaction "{SNK, Titen, TrkB}".

The frequent pattern mining component then takes these transactions and searches for frequent patterns. In this example, it finds that when the gene "SNK" has a high response, the gene "PDGFR-a" usually also has a high response. It also finds that this pattern is only common in the group



Separate rules for separate groups
Include unselected items when creating rules
Include empty transactions when calculating support
Remove Intersection of rules
Support 0.2
Confidence 0.75

```
SNK-->PDGFR-a 0.75
SNK-->Titen 0.75
Titen-->PDGFR-a 0.833
Titen-->SNK 0.75
```

Figure 3. A heat map of patient-gene responses is linked to the relapse/no relapse status of the patients. Selecting a gene response threshold (middle image) creates item sets which are used by a frequent pattern mining component. Patients in the red (non-relapse) group have some gene expression patterns not common in the other group, and selecting one such pattern (bottom image) shows only the genes involved, both in the bar chart with genes and in the heat map.

of patients with no relapse, so we may have found a gene expression pattern indicating low risk of relapse.

Selecting one or more patterns shows only the genes corresponding to the selected patterns, both in the heat map and in the bar chart with genes, and more information can then be retrieved for these. Changing the heat map threshold or the grouping of the patients causes the pattern mining component to redo the pattern mining.

E. Available Components

Currently the following components are available:

Visualization Plugins: *Bar charts*, easily extended to other standard chart types; *Scatter plots*, for numerical or date/time data; *Clocks*, for time data; *Life tables*, survivability charts, usually the number of patients still alive (or still relapse free etc.) as a function of time (commonly used to compare medical treatments); *Heat maps*, intensity maps; *Lines on maps*, for example roads; *Points on maps*, for example building locations; *Linked points on maps*, for example for geographical time series data; *Gene Info*, showing information from and linking to gene databases; *Parallel coordinates*[15]; and *Storygraphs*[16].

Data Source Plugins: *CSV* (comma separated vector) parsing, *XML* parsing, and *Web Service* wrapping.

Analysis Plugins: *Clustering*, supporting several common clustering algorithms, and *Frequent Pattern Mining*.

Conclusions

We described a system for visual exploration of data. It uses direct manipulation of the visualization results in linked views of the data. Adding new data sources is possible at any time (data mash-up) and adding new visualization tools (or clones of ones already used) can also be done at runtime. The system is built using a software component based framework, making it easy to build new components and to prototype new interfaces etc.

The main new extension compared to previous versions of the system is that it is possible to use components that create new complex data at runtime, for example machine learning components or statistical analysis components. These can be linked just like any other component, and changed parameter settings etc. will automatically be reflected in all other linked views (or analysis components) as soon as the new calculations are finished. One example in our system is a frequent pattern mining component that generates rules based on frequent patterns in the data. These rules are generated based on the selections and groupings in other linked components and the resulting rules can then be the base of new groupings or selections (or calculations in other analysis components).

References

[1] J. C. Roberts, "State of the art: Coordinated & multiple views in exploratory visualization," in *Proc. of CMV '07*, Washington, DC, USA, 2007, pp. 61–71.

[2] J. Sjöbergh and Y. Tanaka, "Visual data exploration using Webbles," in *Proc. of the Webble World Summit*, ser. Springer CCIS, vol. 372, Erfurt, Germany, 2013, pp. 119–128.

[3] B. Shneiderman, "The eyes have it: A task by data type taxonomy for information visualizations," in *Proceedings of VL'96*, Washington, DC, USA, 1996, pp. 336–343.

[4] M. Kuwahara and Y. Tanaka, "Webble world – a Web-based knowledge federation framework for programmable and customizable Meme Media objects," in *Proc. of Frontier Computing 2010*, Taichung, Taiwan, 2010, pp. 372–377.

[5] Y. Tanaka, *Meme Media and Meme Market Architecture*. Piscataway; NJ; USA: IEEE Press, 2003.

[6] I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, and T. Euler, "YALE: Rapid prototyping for complex data mining tasks," in *KDD'06: Proceedings of the 12th ACM SIGKDD*, Philadelphia, PA, USA, 2006, pp. 935–940.

[7] M. Livny, R. Ramakrishnan, K. Beyer, G. Chen, D. Donjerkovic, S. Lawande, J. Myllymaki, and K. Wenger, "DEVise: Integrated querying and visual exploration of large datasets," in *Proceedings of SIGMOD'97*, Tucson, AZ, USA, 1997, pp. 301–312.

[8] C. Ahlberg, "Spotfire: An information exploration environment," *SIGMOD Rec.*, vol. 25, no. 4, pp. 25–29, Dec. 1996.

[9] A. Aiken, J. Chen, M. Stonebraker, and A. Woodruff, "Tioga-2: a direct manipulation database visualization environment," in *Proc. of ICDE'96*, New Orleans, USA, 1996, pp. 208–217.

[10] C. North and B. Shneiderman, "Snap-together visualization: a user interface for coordinating visualizations via relational schemata," in *Proceedings of AVI'00*, Palermo, Italy, 2000, pp. 128–135.

[11] M. R. Berthold, N. Cebron, F. Dill, T. R. Gabriel, T. Kötter, T. Meinl, P. Ohl, K. Thiel, and B. Wiswedel, "KNIME - the Konstanz information miner: Version 2.0 and beyond," *SIGKDD Explorations Newsletter*, vol. 11, pp. 26–31, 2009.

[12] J. Demšar, T. Curk, A. Erjavec, Črt Gorup, T. Hočevar, M. Milutinovič, M. Možina, M. Polajnar, M. Toplak, A. Starič, M. Štajdohar, L. Umek, L. Žagar, J. Žbontar, M. Žitnik, and B. Zupan, "Orange: Data mining toolbox in Python," *Journal of Machine Learning Research*, vol. 14, pp. 2349–2353, 2013.

[13] T. Sugibuchi and Y. Tanaka, "Integrated visualization framework for relational databases and web resources," in *Proceedings of IHI'04*, Dagstuhl Castle, Germany, 2004, pp. 159–174.

[14] J. Sjöbergh, M. Kuwahara, and Y. Tanaka, "Visualizing clinical trial data using pluggable components," in *Proceedings of IV'2012*, Montpellier, France, 2012, pp. 291–296.

[15] A. Inselberg and B. Dimsdale, "Parallel coordinates: a tool for visualizing multi-dimensional geometry," in *VIS'90: Proceedings of the 1st conference on Visualization '90*, Los Alamitos, CA, USA, 1990, pp. 361–378.

[16] A. Shrestha, Y. Zhu, B. Miller, and Y. Zhao, "Storygraphs: Extracting patterns from spatio-temporal data," in *Proceedings of IDEA'13*, Chicago, IL, USA, 2013, pp. 96–104.