Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

AW2 Ausarbeitung

Truong Vinh Phan

Interactive Data Visualization

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Contents

1 Introduction

There are now more open data sources than ever before, partly thanks to several popular open data movements. As a result, data sets are growing larger every year with most cases being multivariate. Big data analysis is about extracting hidden properties, trends or patterns from large volumes of unstructured data. One of the more popular approaches to do this is through data visualization. This is one area where exploratory graphics become extremely useful. These types of graphics are defined as being fast and informative rather than slow and precise. They are in most cases used in large quantities, and thus are not intended for presentation.

Visualizing big data by means of exploratory approach requires interactive graphics. Thus, the goal is to make use of traditional and standard graphics in combination with appropriate interactive techniques in order to facilitate exploratory data analysis through interactions. In this report, three approaches to interactive visualization are introduced and discussed.

2 Tree-Maps: A Space-Filling Approach to the Visualization of Hierarchical Information Structures

2.1 Motivation

A large portion of the information around us is hierarchically structured: corporate organizations, internet addressing, library cataloging, computer programs... and so on. Small structures are easy to understand, but this is not the case with larger structures. In this paper, Shneiderman et al. [18] presented an interactive visualization method for hierarchical information called Tree-Maps, which takes advantage of human's visual ability to recognize spatial configuration of elements in a picture and perceive relationships much faster than when they scan text [10].

This method maps hierarchical information to a 2-D rectangular display in a space-filling fashion. It utilizes 100% available space and provides interactive control to specify structural (e.g. depth bounds...) and content (display properties e.g. color mappings...) information. Display space is partitioned into rectangular bounding boxes and can be allocated proportionally to the importance of the information. The drawing of nodes and display size can be interactively controlled by the user, thus allowing for hierarchical structures with large number of nodes to be displayed and manipulated even within a fixed display space.

2.2 The Tree-Maps Method

The main goal is to display the tree structure in a visually appealling way while fully utilizing available space, which is difficult. Interactive elements enable users to control both structural and content representation, e.g. visual properties... to maximize the utility of the visualization.

By controlling the partitioning of the display space, structural information can be adjusted to best fit the task. The representation of Tree-Map displays is similar to those of quad-trees and k-D trees, with the key difference lies in the direction of the transformation. Tree-Maps represents hierarchical structures on 2-D displays, as opposed to quad-trees, which create hierarchical structures to store 2-D images [15]. A weight, which may be single or combined domain properties, e.g. disk usage, is assigned to each node to determine the size of the node's bounding box and can be viewd as a measure of importance [9]. The following relationships should always hold:

1. If *Node1* is an ancestor of *Node2*, then *Node1*'s bounding box is either equal to, or completely encloses *Node2*'s bounding box.
2. The bounding boxes of two nodes intersect if and only if one node is an ancestor of the other.
3. The size of a node's bounding box is strictly proportional to its weight.
4. The weight of a parent node is greater than or equal to the sum of the weights of its children.

Structural information is implicitly presented, but can also be explicitly indicated by nesting, which enables direct selection of all nodes (internal and leaf). The disadvantage thereof is the reduction of displayable tree size [20]. While non-nested displays can only enable direct selection for leaf nodes, a pop-up display can provide further information and facilities. And although non-nested displays also cannot represent internal nodes in degenerate linear sub-paths, such paths rarely occur.

The process of mapping content information to the display can be manipulated through a variety of display properties, which determine how a node is drawn within its bounding box. Aside from primary visual properties such as color, shape, texture, etc. other domain dependent properties may also exist, which result in a rich set of mapping possibilities between content information and display properties [7].

Interactive elements are essential and critical, since the number of variables that can be coded in the tree is limited and there is also an upper bound on the complexity of graphical representation for human perception [8, 11]. Dynamic feedback is available through the use of pop-ups which show information about the current node.

2.3 Algorithms

The Tree-Maps method consists of two algorithms. The first one is the *drawing algorithm* (Fig.1), which functionality is to draw a series of nested boxes representing the tree structure. With this algorithm, the Tree-Map can be drawn during one pre-order pass through the tree in O(n) time under the condition that node properties (weight, name...) have already been pre-computed and assigned.

The second algorithm is the *tracking algorithm* (Fig.2), that enables the path to a node containing a given point in a display to be determined using only a simple descent.

```
DrawTree()      The node gets a message to draw itself      The Root node is set up prior to the original recursive call
{   doneSize = 0;                                           The percent of this nodes subtree drawn thus far
    PaintDisplayRectangle();                                The node sends itself a Paint Message
    switch (myOrientation) {                                Decide whether to slice this node horizontally or vertically
        case HORIZONTAL:
            startSide = myBounds.left;                      Set start for horizontal slices
        case VERTICAL:
            startSide = myBounds.top;                       Set start for vertical slices
    }
    if (myNodeType == Internal) {                           Set up each child and have it draw itself
        ForEach (childNode) Do {
            childNode->SetBounds(startSide, doneSize, myOrientation); Set childs bounds based on the parent partition taken by previous
                                                                      children of parent
            childNode->SetVisual();                         Set visual display properties (color, etc.)
            childNode->DrawTree();                          Send child a draw command
}}}

SetBounds(startSide, doneSize, parentOrientation)
{   doneSize = doneSize + mySize;                           How much of the parent will have been allocated after this node
    switch (parentOrientation) {                            Decide which direction parent is being sliced
        case HORIZONTAL:
            myOrientation = VERTICAL;                       Set direction to slice this node for its children
            endSide = parentWidth * doneSize / parentSize;  How much of the parent will have been sliced after this node
            SetMyRect(startSide + offSet,                   Left side, Offset controls the nesting indentation
                parentBounds.top + offSet,                  Top
                parentBounds.left + endSide - offSet,       Right
                parentBounds.bottom - offSet);              Bottom
            startSide = parentBounds.left + endSide;        Set start side for next child
        case VERTICAL:
            myOrientation = HORIZONTAL;                     Set direction to slice this node for its children
            endSide = parentHeight * doneSize / parentSize;
            SetThisRect(parentBounds.left + offSet,         Left side
                startSide + offSet,                         Top
                parentBounds.right - offSet,                Right
                parentBounds.top + endSide - offSet);       Bottom
            startSide = parentBounds.top + endSide;         Set start side for next child
}}
```

Figure 9. Drawing Algorithm

Figure 1: Drawing algorithm.

2.4 Conclusion

The space-filling approach to visualization of hierarchical structures, as represented by the Tree-map algorithm, has great potential for exploratory visualization of large data sets. First,

```
FindPath(point thePoint)
{   if node encloses thePoint then
    foreach child of thisNode do {
        path = FindPath(thePoint);
        if (path != NULL) then
            return(InsertInList(thisNode, path));          Add child to path
    }
    return (NULL);                                         Start path, thePoint is in this node, but not in any of its children
}
```

Figure 10. Tracking Algorithm

Figure 2: Tracking algorithm.

it enables effective representation of large hierarchies in a limited space, which is often the challenge when visualizing large data set with today standard displays. It also has much space for future extensions, such as an alternate scheme for structural partitioning, for visual display of various content information, including numeric and non-numeric. Tree-map visualization could be enhanced further with dynamic views, e.g. animated time slices and more advanced operations on elements of the hierarchy, besides the standard ones, such as zooming, panning, selecting, searching, etc. Because large data sets often contain some sorts of hierarchy, this visualization technique can be applied in a wide variety of applications, e.g. in financial portfolios.

3 Trellis Displays for High-Dimensional Data Visualization

3.1 Definition

Becker et al. [2] introduced Trellis displays to visualize multivariate data. Unlike mosaic-plots, which use a recursive layout, trellis displays use a grid-like (lattice) layout to arrange conditioned plots onto panels. The use of the same scales in all panel plots enables plot comparison across rows and columns.

A single trellis display can theoretically hold up to seven variables. Five of them need to be categorical, with up to three can be used as conditioning variables to form rows, columns and pages. The other two can be continuous. Up to two variables, called axis variables, can be plotted in a panel plot. All the panel plots share the same scale. Rows, columns and pages of the trellis display are created using up to three conditioning categorical variables.

Shingling is a concept which was introduced with trellis displays and is defined as the process of converting a continuous variable into a discrete one by splitting a continuous variable into (overlapping) intervals. Overlapping intervals may lead to various data representations in a trellis display. As an example of a more complex trellis display, Fig.3 shows a cars data set being plotted using scatterplots of Gas Mileage (MPG) vs. Weight (axis variables). The

grid is formed by the conditioning variables Car Type (x-axis) and Drive (y-axis). The adjunct variable Number of Cylinders is color-coded in the scatterplots.

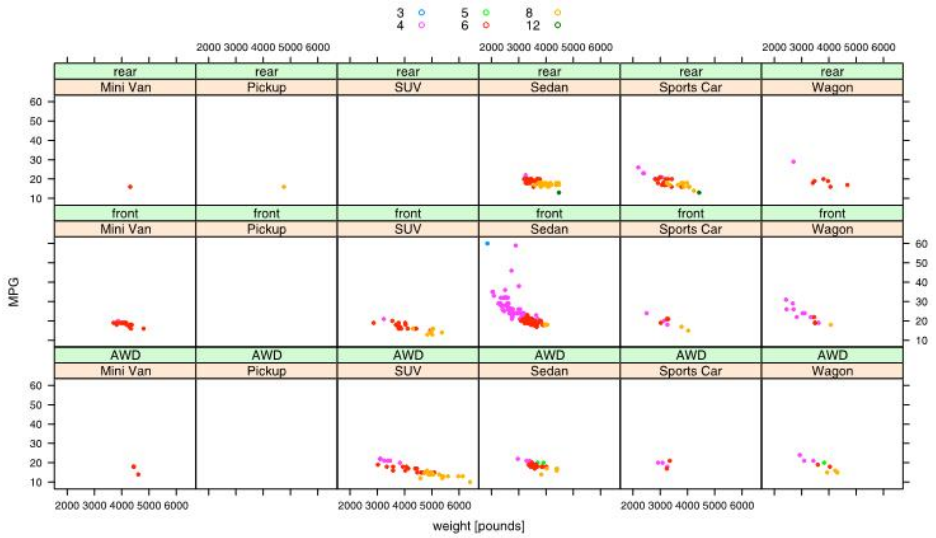

Figure 3: A trellis display with five variables.

3.2 Trellis Displays with Interactive Elements

Based on the fact that the conditional framework and the single view in a panel of a trellis display can be seen as static snapshots of interactive graphics and highlighted part of the panel plot graphics for the conditioned subgroup, respectively, trellis displays can be extended with interactive elements. Fig.4 shows some of the possible interactions in an interactive session, including selecting a specific subgroup in the left mosaic plot or moving the brush (selection indicator) along one or two axes of the plot. This is where the concept of shingle variables can be useful, as the selection from the brush can be regarded as an interval of a shingle variable. The shingling process divides a continuous variable into intervals, which correspond to the snapshots of the continuous brushing process, as illustrated in Fig.5

3.3 Conclusion

Trellis displays are most suitable for continuos axis variables, categorical conditioning variables/adjunct variables. Large data sets are most likely to be multivariate, for which multivariate visulization techniques like trellis displays are designed. Despite the advantage of a flat learning curve and the ability to add model information to the plots, current trellis display implementations do not offer any interactions. Still, exploratory data analysis can be facilitated

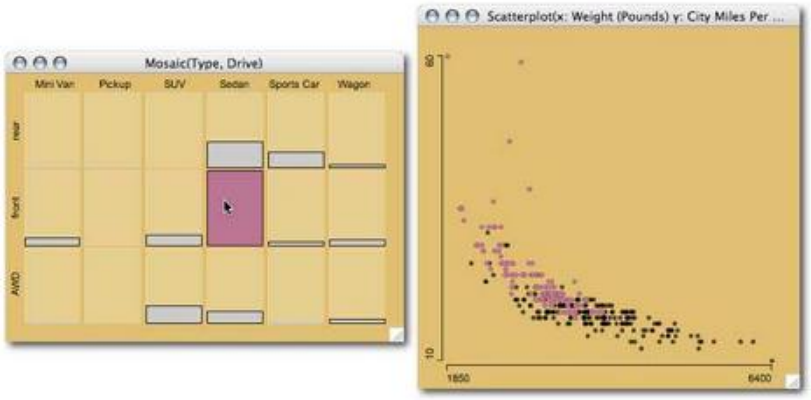

Figure 4: Selecting a subgroup in the multiple-barchart view (left) highlights the subgroup in the corresponding panel plot (right)

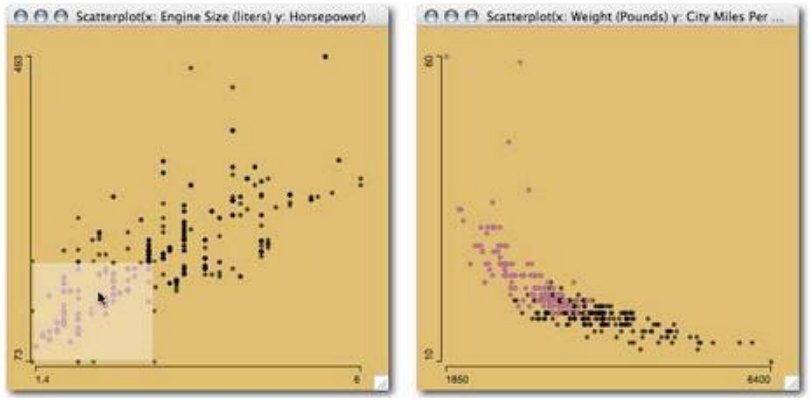

Figure 5: Brushing the scatterplot (left) highlights the corresponding subgroup in the panel plot (right)

with the use of interactive linked graphics, e.g. linking panel plots to barcharts/mosaicplots of the conditioning variables, brushing over shingle variables, etc.

4 Linked Views for Visual Exploration

4.1 Introduction

One of the most common challenges in visualization is the physical dimensional limitation of the presentation device, whether it be paper or computer screen, visualization is limited in a 2-D space. To address this problem, there are typically four approaches:

- Use of a virtual reality/pseudo 3-D environment in a 3-D setting to portray higher-dimensional data.
- Projection of high-dimensional data onto 2-D coordinate system using data reduction methods.
- Use of a non-orthogonal coordinate system, such as parallel coordinate.
- Linking of multiple low-dimensional displays, which is discussed here.

This idea is not new, with the use of identical plot symbols and colors to indicate the same cases across multiple displays in the development of static displays as mentioned in [21] and [6] and first implemented in [12] to connect two scatterplots. The most widely used implementation of linked views is the scatterplot brushing, including linking in both scatterplots and scatterplot matrices, as promoted in [3].

The main benefits of using linked views with regard to exploratory data analysis are the simplicity of underlying graphical displays, speed and flexibility in portraying various data aspects. Another advantage of linked views is the applicability to complex data structures, such as geographically referenced data in the context of spatial data exploration, as discussed in [1], [23] and [14]. Linked views is mainly applied in statistical exploration of datasets, to address issues such as finding unusal behaviors, detecting relationships, patterns, etc.

4.2 Linking Schemes and Structures

The principal behind linked views is the sharing and exchanging information between plots. To achieve this, first a linking mechanism is needed to establish a relationship between the plots, then two questions need to be answered, namely, which information is shared and how? A separation of data displays in their components, as proposed by Wilhelm in [22] set the foundation to explore a wide variety of linking schemes and structures. According to [22], a display D is made of a frame F, a type with a set of graphical elements G and a set of scale S_G , a model X with scale S_X, and a sample population Ω. Thus, the data part is the pair $((X,S_X),\Omega)$ and the pair $(F,(G,S_G))$ is the plotting part.

According to the above definition, it is theoretically possible to define a linking structure as a set of relations among any two components of the displays. In practice, however, only the relations between identical layers of the display are of relevance. Thus, possible linking schemes between active display D_1 and passive display D_2 are as depicted in Fig.6

From the separation of data display in components, linking schemes can be categorized into four types:

- Linking sample populations: defined as a mapping $m : \Omega_1 \rightarrow \Omega_2$ in which elements of sample population space Ω_1 are mapped to some elements of space Ω_2. There

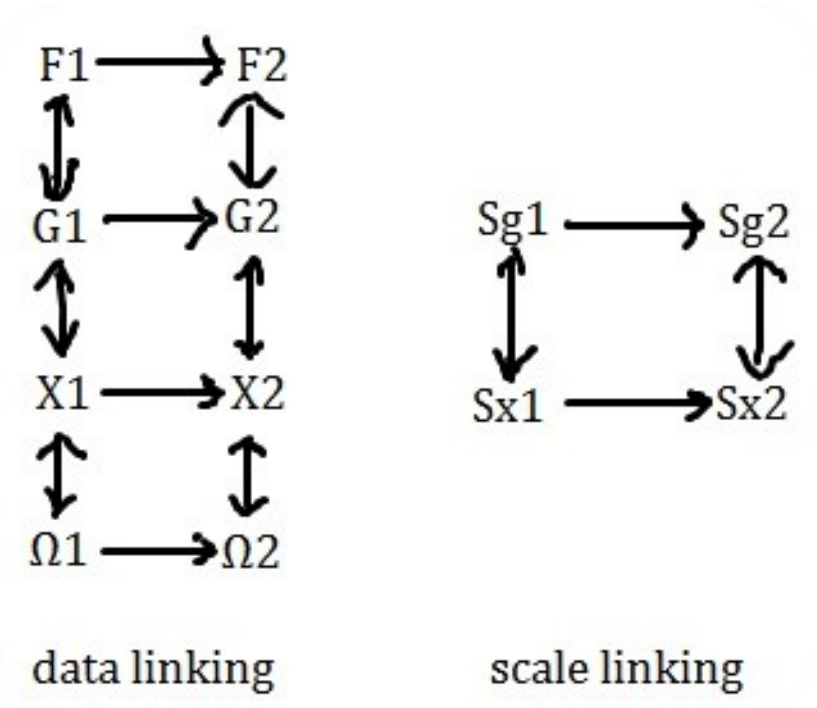

Figure 6: Possible linking schemes between sender plot D_1 and receiver plot D_2.

are three common cases of linking sample population: *Identity linking* (empirical linking : $id : \Omega \to \Omega$), *Hierarchical linking* ($m : \Omega_1 \to \Omega_2$ with filtration), and *Neighborhood/Distance linking* (for geographical data, linking relation depends on definition of neighborhood/distance).

- Linking models: models describe precisely the amount of information to be visualized. For example, the histogram of a quantitative variable is based on the categorization model. Linking models can be further categorized into type linking and scale linking, with scale linking being the more prominent case and most widely implemented in the form of sliders for dynamic queries. In [16] and [17], Shneiderman provides a good overview about this implementation of scale linking. Linking observations is restricted to the variables used in the model, as illustrated in Fig.7. Young et al. have provided a fairly comprehensive discussion and proposals about linking observations in [24].

- Linking types: the type layer covers most visible components in a graphical display and aims at representing the model as well as possible. Due to this close connection, congruities at the type level typically are the result of linked models. Direct link between type levels without model linking is uncommon, except for color and size, which are attributes that can be linked regardless of model linking. It is often required to link type information to properly compare between various plots.

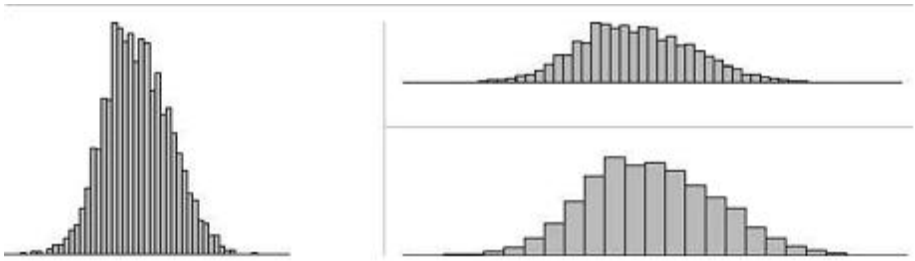

Figure 7: Three histograms for the same variables. Two plots (right) have same frame size, different scales. Plot in top right has the same scale as plot on the left.

- Linking frames: frames control the shape and size of the plot window. Linking frames is important for the correct comparison of graphical displays and to achieve a screen space-saving layout.

4.3 Implementation Techniques for Linked Views

Information sharing is the back-end mechanism that drives the linked views paradigm. Information sharing occurs in various circumstances, such as in an interactive session with the user making changes to a plot while exploring and investigating the data. This scenario raises the question of where the information should go and how it can be optimal represented. According to Robert et al., three different strategies for realising linked views can be distinguished: [13]

- Replacement strategy: While this strategy can be applicable for plot parameters, it proved to be not useful for subsetting and conditioning approach because of the replacement of old information, except for the case in which each observation has its individual plot symbols. Even then, the inability to compare different versions of the plot makes this strategy inappropriate for exploratory data analysis, where it is essential to keep track of changing scenarios and different plot versions. As introduced in [14], implementing a history system similar to those in geovisualization systems to help keep track of plot changes is very helpful.
- Overlaying: While this strategy is typical for comparing two conditional distributions, it creates two problems: one is the basic restriction in the freedom of parameter choice for the selected subset, because the parameters are inherited from the original plot, the other is the problem of overplotting/occlusion, in which part of the original display is hidden by the overlaid plot. This problem is mostly irrelevant for area-based displays and scatterplots but plays an important roles in complex plots such as boxplots. Fig.8 shows the overlaying strategy in a scenario of a histogram (left) being linked to a barchart (right). The selection in the barchart is propagated to the histogram and

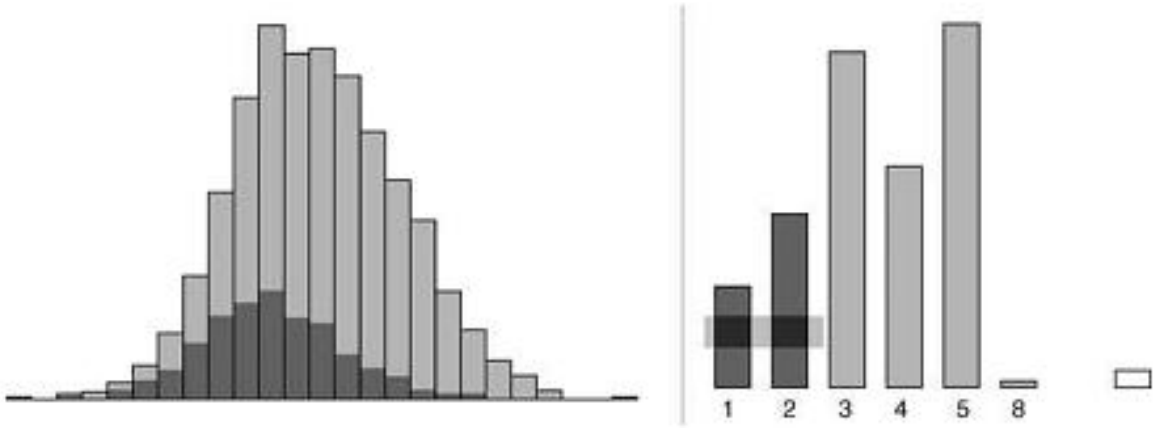

Figure 8: Overlaying strategy in linked plots.

overlaid on the original plot. Plot parameters are inherited.

- Repetition: The third strategy of visualizing linked views is to multiply the displays , with each display represents a different view of the data and all are presented to the user at the same time. The advantage is that the user gets a comprehensive picture of the data, a fairly complete overview which enables easy observation of the impact of parameter changes/user interactions. The downside is that the overview might become complex by various changing and adapted views, therefore a mechanism to keep track of various changes and user interactions, as well as an effective system to arrange the displays on computer screen are essential. Juxtaposition is an example form of the repetition strategy that works very effectively for subsetting scenarios.

4.4 Special Forms of Linked Views

More complex forms of linking such as m-to-1 in hierarchical linking poses a few challenges. Take, for example, two levels of a hierarchy: a macro level (e.g. a set of counties/states), and a micro level (e.g. a set of cities/towns). A partial selection of some cities/towns would be represented best by partial highlighting. The problem arises when the macro level is represented by non-regular shapes that cannot be subdivided properly. A general approach to this problem would be to use different color intensities to fill the according graphical element, which is recommended for non-regular shaped graphical elements as well as other shapes for its easiness of decoding. This approach is illustrated in Fig.9

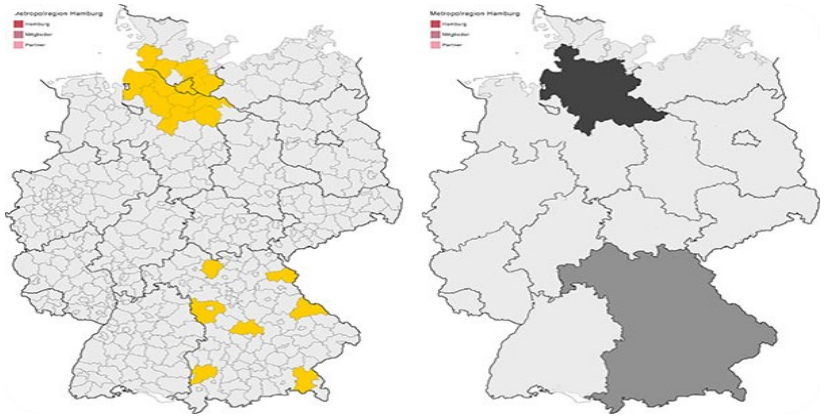

Figure 9: Visualization of a hierarchical linking scenario with two maps. The map on the right shows the states in Germany and on the left is a more detailed map showing counties. The intensity of the filled color of the states depends on the number of selected counties.

4.5 Conclusion

Linking multiple simple 2-D views by establishing relationships between plots that show different aspects of related data enables the user to explore and understand structures and patterns of more complex datasets. This concept is essential in the field of visual data mining and provides the required HCI to understand hidden structures and patterns. The linking procedures work best with complex datasets, as in the case of big data, which have very large number of observations and/or variables (high-dimensional), a mixture of variable types, as well as possibly incomplete and/or missing values. Generalization of linking and the use of a same scale ensure consistency in data views and comparisons of visual displays.

5 Summary

The papers in this report describe approaches to addressing the challenge of visualizing multivariate data enhanced with interactive components in exploratory data analysis settings. With data today having become simply too large and often have too short a lifespan, more problems and challenges surface. The above visualization techniques have been around for quite some time, therefore didn't take into account one of the main problem of big data: data quality, for example imperfection, defects, distortions, gaps, etc. in data entries. An effective visualization system must therefore make users aware of the quality of the data by explicitly conveying data quality attributes, besides data content. The data itself must also undergo data cleansing processes before being visualized.

With today's powerful displays being widely available and the continuous progress in display technologies, in order to achieve effectiveness and usability, interaction designs and web technologies should be incorporated into visualization systems to make good use of display resources, including mobile displays. A wide variety of research papers have been dedicated to solving these challenges, for example in [25], Zaixian et al. address the data quality issue in multivariate data visualization. A system designed to deliver visualization to mobile displays through web-based OLAP is described in [19] and finally in [4], Johanna et al. describe a system for interactive exploration of petascale volume biological data, taking into account the incompleteness of data and scalability.

References

[1] ANSELIN, L.: Interactive techniques and exploratory spatial data analysis. In: *Geographical Information Systems: Principles, Techniques, Management and Applications* (1999)

[2] BECKER, R. ; CLEVELAND, W. ; SHYU, M.: The visual design and control of trellis displays. In: *Journal of Computational and Graphical Statistics* 6 (1996), Nr. 1, S. 123–155

[3] BECKER, R. A. ; CLEVELAND, W. S. ; WILKS, A. R.: Dynamic graphics for data analysis. In: *Statistical Science 2* (1987)

[4] BEYER, J. ; HADWIGER, M. ; AL-AWAMI, A ; JEONG, Won-Ki ; KASTHURI, N. ; LICHTMAN, J.W. ; PFISTER, H.: Exploring the Connectome: Petascale Volume Visualization of Microscopy Data Streams. In: *Computer Graphics and Applications, IEEE* 33 (2013), July, Nr. 4, S. 50–61. – ISSN 0272-1716

[5] CHEN, Chun houh ; HÄRDLE, Wolfgang ; UNWIN, Antony: *Handbook of Data Visualization*. Berlin-Heidelberg : Springer-Verlag, 2008

[6] DIACONIS, P. N. ; FRIEDMAN, J. H.: *M and N plots in Recent Advances in Statistics*. 1983

[7] DING, Chen ; MATETI, Prabhaker: A framework for the automated drawing of data structure diagrams. In: *IEEE Transactions on Software Engineering* (1990)

[8] ELLSON, Richard: *Visualization at work*. 1990

[9] FURNAS, George W.: Generalized fisheye views. In: *Proceedings of ACM CHI'86 Conference on Human Factors in Computing Systems, Visualizing Complex Information Spaces* (1986)

[10] KAMADA, Tomihisa: *On Visualization of Abstract Objects and Relations*. 1988

[11] KUHN, Werner: Editing spatial relations. In: *Proceedings of the 4th International Symposium on Spatial Data Handling* (1990)

[12] MCDONALD, J. A.: *Interactive graphics for data analysis*. 1982

[13] ROBERT, J. C. ; KNIGHT, R. ; GIBBINS, M. ; PATEL, N.: *Multiple Window Visualization on the Web using VRML and the EAI*. 2000

[14] ROBERTS, J. C.: *Exploratory visualization with multiple linked views*. 2004

[15] SAMET, Hanan: *Design and Analysis of Spatial Data Structures*. Addison-Wesley, 1989

[16] SHNEIDERMAN, B.: *Dynamic queries for visual information seeking*. 1994

[17] SHNEIDERMAN, B.: *Information Visualization: White Paper*. 1997. – URL http://www.cs.umd.edu/hcil/members/bshneiderman/ivwp.html

[18] SHNEIDERMAN, Ben ; JOHNSON, Brian: *Tree-Maps: A Space-Filling Approach to the Visualization of Hierarchical Information Structures*. 1991

[19] TIM, H. ; LUK, Wo-Shun ; STEPHEN, P.: Data Visualization on Web-based OLAP. In: *ACM* (2011)

[20] TRAVERS, Michael: A visual representation for knowledge structures. In: *ACM Hypertext '89 Proceedings, Implementations and Interfaces* (1986)

[21] TUFTE, E. R.: *The Visual Display of Quantitative Information*. 2nd. Cheshire : Graphics Press, Mai 2001

[22] WILHELM, A. F. X.: Interactive statistical graphics: The paradigm of linked views. In: *Handbook of Statistics* 24 (2005)

[23] WILLS, G.: *Spatial Data: Exploration and Modelling via Distance-Based and Interactive Graphics Methods*. 1992

[24] YOUNG, E. W. ; FALDOWSKI, R. A. ; MCFARLANE, M. M.: Multivariate statistical visualization. In: *Handbook of Statistics* 9 (1993)

[25] ZAIXIAN, X. ; SHIPING, H. ; MATTHEW, O. W. ; ELKE, A. R.: Exploratory Visualization of Multivariate Data with Variable Quality. In: *IEEE Symposium on Visual Analytics Science and Technology* (2006)