VisBricks: Multiform Visualization of Large, Inhomogeneous Data

Alexander Lex, Hans-Jörg Schulz, Marc Streit, Christian Partl, and Dieter Schmalstieg

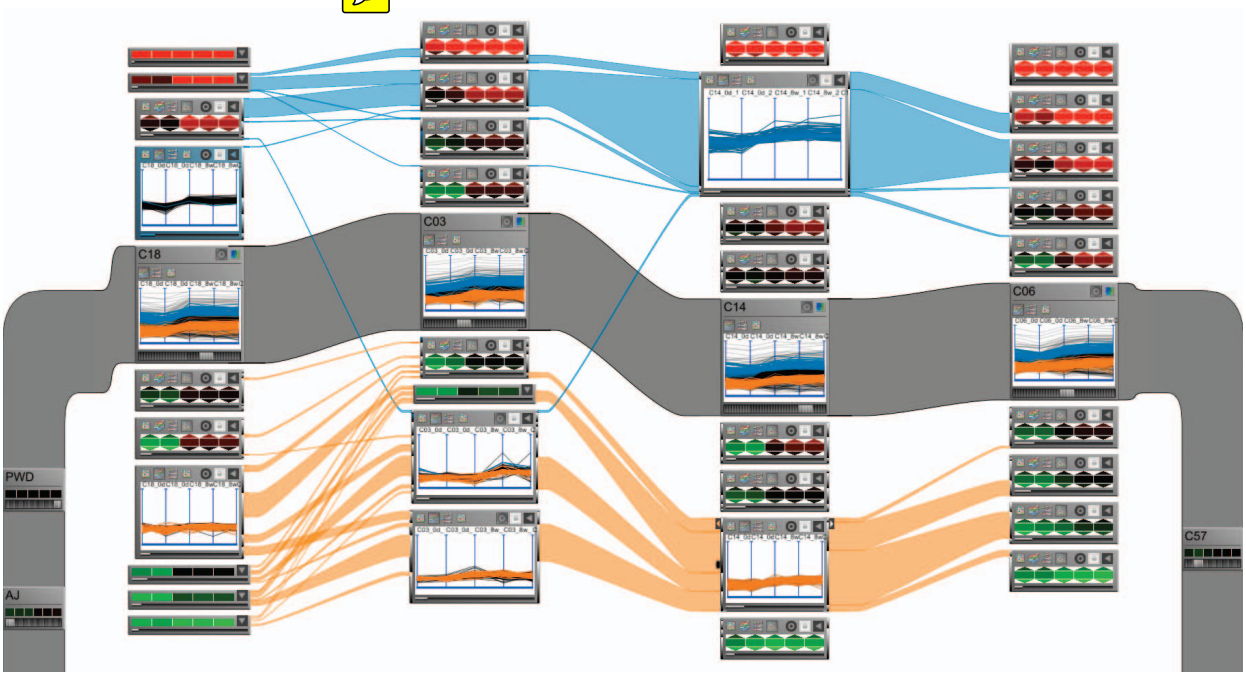


Fig. 1. *VisBricks* in action: Four different groups of dimensions with different numbers of clusters per group. The gray arch connects the overviews of the groups. The arches show how the data is distributed over the clusters in that group, thus summarizing the specifics of a dimension group. The clusters themselves are shown in stacked VisBricks above and below the arch depending on whether their average data values are higher or lower than the overall average for the group. Colored ribbons indicate how data items are distributed across clusters of multiple dimension groups.

Abstract—Large volumes of real-world data often exhibit inhomogeneities: vertically in the form of correlated or independent dimensions and horizontally in the form of clustered or scattered data items. In essence, these inhomogeneities form the patterns in the data that researchers are trying to find and understand. Sophisticated statistical methods are available to reveal these patterns, however, the visualization of their outcomes is mostly still performed in a one-view-fits-all manner. In contrast, our novel visualization approach, *VisBricks*, acknowledges the inhomogeneity of the data and the need for different visualizations that suit the individual characteristics of the different data subsets. The overall visualization of the entire data set is patched together from smaller visualizations, there is one VisBrick for each cluster in each group of interdependent dimensions. Whereas the total impression of all VisBricks together gives a comprehensive high-level overview of the different groups of data, each VisBrick independently shows the details of the group of data it represents. State-of-the-art brushing and visual linking between all VisBricks furthermore allows the comparison of the groupings and the distribution of data items among them. In this paper, we introduce the VisBricks visualization concept, discuss its design rationale and implementation, and demonstrate its usefulness by applying it to a use case from the field of biomedicine.

Index Terms—Inhomogeneous data, multiple coordinated views, multiform visualization.

1 Introduction

Data from real-world applications is often inhomogeneous, exhibiting sparse and non-uniform distributions across a vast, multi-dimensional data space. The main challenge posed by this situation is that different subsets of an inhomogeneous data set need to be treated differently, *i.e.*, stored differently, queried differently and shown differently. For instance, methods that work well in numerical regions of the inhomogeneous data space may not be suitable for areas containing categorical data. Common analytical solutions, such as *OLAP* (online analytical processing) or hierarchical clustering, segment the data space accordingly to store and process the individual, more homogeneous parts in a way that is better suited to their specifics. It only seems natural to extend this idea of an independent and customized treatment of each part of the inhomogeneous data from analytics to their graphical representation. Although in some areas, such as graph visualization, this idea is actively pursued, for tabular, multi-dimensional data, minimal research has been published in this direction, which is surprising, as Thomas' & Cook's Visual Analytics research agenda from 2005 states that "new representations are needed to help analysts understand complex heterogeneous information spaces" [30].

The *VisBricks* visualization approach presented in this paper aims

• *A. Lex, H.-J. Schulz, M. Streit, C. Partl, and D. Schmalstieg are with the Graz University of Technology, Austria, E-mail: lex—schulz—streit—partl—schmalstieg@icg.tugraz.at.*

Manuscript received 31 March 2011; accepted 1 August 2011; posted online 23 October 2011; mailed on 14 October 2011.
For information on obtaining reprints of this article, please send email to: tvcg@computer.org.

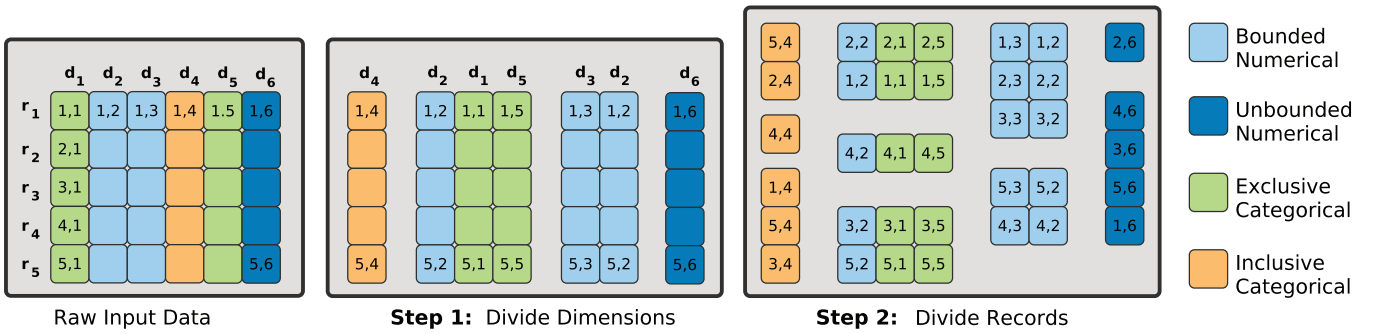


Fig. 2. The initial data matrix may encompass several different data characteristics (color-coded) and value distributions within each dimension's (column's) data range. Thus, the data matrix is first broken up into more homogeneous groups of dimensions. Depending on which homogeneity criteria are used, this may still result in dimension groups with dimensions of different characteristics and even in dimensions being grouped into multiple dimension groups. Then in the second step, these dimension groups are clustered to reveal inhomogeneities within the set of data records. Depending on which cluster algorithm is used, records can be assigned to multiple clusters.

to provide such a new representation in the form of a highly configurable framework, that is able to incorporate any existing visualization as a building block. This method carries forward the idea of breaking up the inhomogeneous data into groups, *i.e.*, vertically into correlated dimensions and horizontally into clusters of records, to form more homogeneous subsets, which can be visualized independently and thus differently. We call such an independent unit a VisBrick. Putting these independent visualizations of data subsets back together creates a so-called *multiform* visualization [25], which gives an overview of the topology of the entire data set by showing groups of related dimensions and clustered data items. The visualization technique embedded in each VisBrick, as well as the overall arrangement of VisBricks, can be tailored to different analysis tasks. Together with a rich set of interactions and visual cues that help to merge, split, rearrange, and reconfigure the VisBricks, this flexible new representation supports many explorative and comparative tasks that otherwise would be difficult to accomplish. A visual impression of an implementation of the *VisBricks* approach is given in Figure 1.

We evaluate our solution with a real world data set from the field of biomedicine. The results are promising and also indicate directions for future research and possible improvements.

2 Preliminaries

Because inhomogeneity in data is multi-faceted, it is necessary to establish the terms and different notions of inhomogeneity.

Data used in analytical tools is most commonly rectangular in nature, consisting of rows and columns [15]. Tables and matrices are common instances of this type of data organization. In the following, we refer to such a rectangular data set as a data matrix M with a set of dimensions D as columns and a set of records R as rows. As illustrated in Figure 2, each cell $(i, j) \in M$ contains a value v_{ij} belonging to the i-th record r_i and lying within the value range of the j-th dimension d_j. In this paper, we only consider static data, so that the number of records $|R|$ and the number of dimensions $|D|$ are known and not subject to change over time.

Inhomogeneity/homogeneity is a fundamental property of such data that can be observed vertically on the set of dimensions and horizontally on the set of records. We distinguish inhomogeneity from the slightly different notion of *data diversity*. The latter defines high diversity as corresponding to an even distribution of values [24], which is a property of a rather homogeneous data set. Furthermore, we use the term *inhomogeneity* instead of *heterogeneity* because, in most of the literature, inhomogeneity is defined for a single data set, whereas heterogeneity usually refers to multiple data sets. However, inhomogeneity in a single data set can, of course, be the result of a data fusion of a number of heterogeneous data sets.

In principle, three different sources of inhomogeneity within a data set can be discriminated:

- **semantics** – of different meanings: the more unrelated the data is in terms of meaning, the more inhomogeneous it is
- **characteristics** – of different types: the more inconsistent the data types and value ranges, the more inhomogeneous they are
- **statistics** – of different behaviors/distributions: the less the data is distributed over a value range, the more inhomogeneous it is

The relevance of the three levels of inhomogeneity for dimensions and records is explained in the following.

Inhomogeneous Dimensions: In terms of **semantics**, inhomogeneities can often be found between dimensions with no inherent connection on the level of what they are meant to encode. For example, the columns "first name" and "last name" would belong together because they compose the information "name" and the columns "street", "city", and "zip code" form the information "address". However, "first name" and "zip code" are semantically unrelated. Such groupings are not obvious from a data or meta-data level and have to be specified by the user employing common knowledge.

The dimensions' **characteristics** basically detail a dimension's type, of which we distinguish four: **bounded numerical**, **unbounded numerical**, **exclusive categorical**, and **inclusive categorical**. An example of inhomogeneity between different dimensions would be two bounded numerical types with very different bounds given, *e.g.*, $[0 \dots 1]$ and $[10^6 \dots 10^7]$, which are very hard to analyze together, numerically or visually. The same is the case for dimensions of exclusive categorical data, such as gender, which is an either-or category, and inclusive categorical data, such as professional memberships in, for example IEEE, ACM or Eurographics. Such characteristics can be interactively defined [23] or given in a standardized format such as qnch[1].

Statistics, in contrast, are derived directly from the data using methods such as correspondence analysis, which will determine dependent dimensions that are likely to belong together because the values are correlated.

Inhomogeneous Records: Similar to dimensions, records can be affected by **semantic** inhomogeneity, which is given by external knowledge. This occurs frequently for categorical values; *e.g.*, the professions "high school teacher" and "university professor" relate more to one another than to "restaurant chef", because they both belong to the educational sector. Again, this knowledge is not present in the data itself and has to be provided by the user or through an ontology.

Inhomogeneities stemming from a record's **characteristics**, can be, for example, missing or undefined values. Undefined values are present, but outside of a dimension bound given by the meta-data. Observation of these inhomogeneities is important; these records need to

[1] http://qnch.org

be set aside because they cannot be analyzed together with the regular records. However, their communication is nevertheless important for the analysis [6].

Inhomogeneities uncovered via **statistical** methods such as clustering occur if the data records are distributed unevenly and thus form clusters at certain points or intervals of the overall value range. Data records that have been assigned to the same cluster are thus more alike and form a more homogeneous group of data with respect to the similarity measure used for clustering.

Generally, it is easier to deal with homogeneous data than with inhomogeneous data in terms of computation and visualization: computational and visual analysis do not have to fall back on the level of the least common applicable method that is able to handle all the different types of data on different value ranges and with different underlying meanings. Instead, if the inhomogeneous data set is divided into more homogeneous subsets, *i.e.*, by grouping dimensions and records, the data can be adequately analyzed using methods that are specifically tailored to them. Additionally, vertical and horizontal subdivision can be used together through a sequential subdivision in two steps, as schematically shown in Figure 2. It should be noted that neither of these subdivisions must generate disjointed groups. Instead, it is often sensible to include the same dimension/record in multiple groups, *e.g.*, to achieve meaningful groupings for inclusive categorical values, which allow a record to belong to multiple categories at once.

After the subdivision, each subset of data can be processed and displayed individually according to its properties. However, the individual visualizations are only of limited use if they are not displayed in the context of the overall data set, and thus in the context of all other individual visualizations. Only then do comparison tasks and the analysis of distributions become possible. Thus, the contextualization of the subsets corresponds to the conquer step of a classical divide-and-conquer approach. As the following section will show, many existing approaches for inhomogeneous data share the divide-and-conquer methodology as a fundamental principle.

3 Related Work

Inhomogeneity in data has been investigated most frequently for graph data and its visualization, possibly because of the unfavorable runtime complexities of analysis and layout algorithms for general graphs. Graph layout algorithms benefit greatly from a subdivision of the data into smaller subsets, which can be efficiently processed individually, and then can be compiled for an overall result. In addition to the gain in speed, this strategy can also generate more expressive representations because the sub-layouts can be optimized.

The following two sections give a short overview of the existing approaches for the visualization of inhomogeneous graph and tabular data by discussing different methods that are often used to perform the divide and the conquer steps.

3.1 Dividing Inhomogeneous Data

For large graphs, the subdivision of inhomogeneous data is performed purely in the data space, as it has to be performed before the mapping (*i.e.*, creating the layout), which may be time-consuming. Graph theoretical methods are used to determine more coherent subgraphs within the inhomogeneous overall data set. These subdivision methods are, in most cases, hierarchical clustering algorithms or traversal strategies for identifying connected components; both are often used together. A possible way to combine these methods is to first perform a quick traversal to identify (bi-) connected components that are then further clustered hierarchically in a second step [1].

For multivariate tabular data, statistical subdivision methods are usually employed. In the case of horizontal subdivision, the subdivision is based on the statistics via (hierarchical) clustering, or on the semantics, as is often observed for OLAP-like partitioning of the data space into different value ranges. The latter does not just perform equidistant partitioning, *e.g.*, a person's age in sets of 10; instead, it brings in common knowledge and makes more meaningful partitions, such as being of legal age at 18 or retiring at 65. The same is true for the vertical subdivision, which is based on statistics through the aforementioned correspondence analysis or on grouping dimensions according to their semantics; a user would likely place zip codes and a person's age in different dimension groups, even if for some reason the statistics found a correlation between both.

The divide step is a crucial one, because it pre-determines many of the features a user will later see in a visualization of its results. A falsely parameterized statistical algorithm may result in an utterly useless visualization that does a good job at communicating false results that are not actually representative of the data. Hence, different tools and frameworks have been devised to support the user during the divide step. For vertical subdivision, a hierarchical dimension management framework [33] can be used to construct subspaces, orders and filter dimensions. For the horizontal subdivision of the data, there is, for example, the Hierarchical OLAP visualization [21], which supports the horizontal subdivision of the data space via OLAP and allows the user to interactively steer the subdivision process. Alternatively, the Matchmaker technique [19] allows the user to compare different statistical subdivisions and thus decide on the most plausible one.

It is important to note that the created subdivisions do not necessarily need to be disjointed, even though often they are generated such that they do not have overlaps, which makes the following conquer step easier.

3.2 Conquering Inhomogeneous Data

After the inhomogeneous data has been subdivided into groups, the groups are processed and visualized individually. Finally, the outcomes for all the groups have to be fused together to form a visualization for the whole data set again. The result of this can be a uniform visualization, in which all individual visualizations are of the same kind, or a multiform visualization, in which entirely different visualizations are merged together [25]. In the field of graph visualization, an example of a uniform visualization is the TopoLayout [2], which hierarchically combines different layouts of the subgraphs, but all of the layouts are of the node-link type. For multiform visualizations, there are examples of pairwise combinations of all three major graph representation types, *i.e.*, matrix, node-link, and implicit layout: NodeTrix [12] combines a matrix with a node-link layout; Elastic Hierarchies [34] combine a node-link with an implicit layout; and Rufiange *et al.* [26] combine a matrix with an implicit layout.

Conceptually, there are two ways of assembling an overview of a subdivided tabular data set by patching together the individual visualizations of the subsets. The first possibility is a very rigid arrangement of the visualizations in a certain style that reveals relationships merely by thoughtful positioning of the individual views. Examples for this approach, however, are scarce. Two notable techniques that apply this approach are portals, as used in the DataSplash system [32], and Multiform Matrices [20]. Portals are locally embedded smaller visualizations in a larger base visualization. The relationship among different portals is automatically communicated through their positions, as can be seen in Data Splash, in which the individual visualizations are put in the context of a map representation, and in Multiform Matrices, where the visualizations are placed in a matrix arrangement, clearly conveying which dimensions are shown in which visualization. In theory, both of these existing techniques have the potential to employ multiform visualizations; however, the existing examples of embedded portals always show the same visualization in all portals.

The second possibility is to allow a more flexible arrangement of views and to use visual links to communicate their relationships. An example for a visualization technique using this approach are Parallel Sets [16], which are able to combine multiple bar charts for different categories in a layout connecting the related bar charts via ribbons. Again, although in theory it would be possible to use this technique as a multiform visualization with different views being connected, in practice it has so far only been used in a uniform manner with all views utilizing the same kind of representation. The underlying idea of maintaining the contextual relationship of subsets of data via visual links is an often-employed mechanism. The work by Seo and Shneiderman [28] is an example, in which the associations between

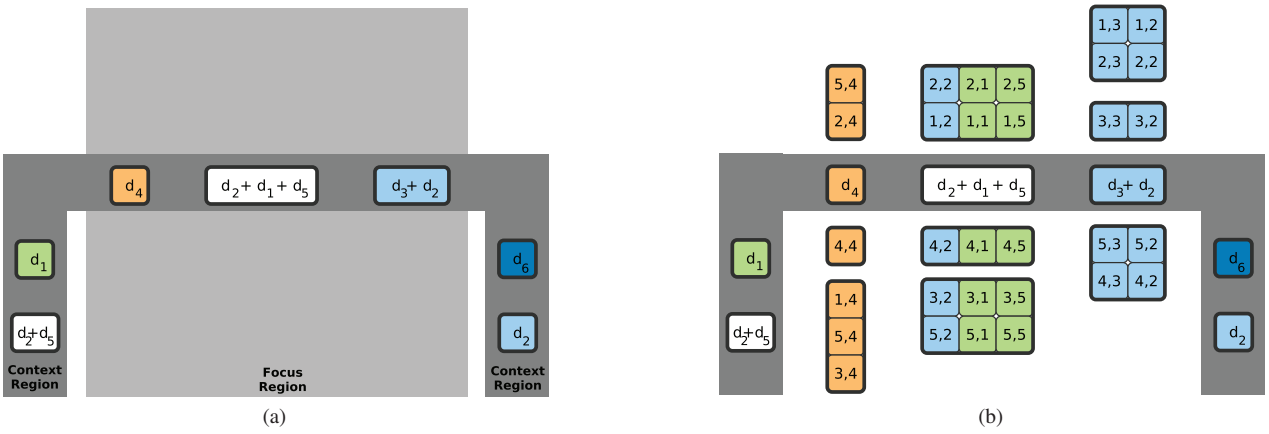


Fig. 3. Basic VisBricks concept. (a) The dimension arch containing the dimension groups with the focus region and the context regions in the legs. Dimension groups can be homogeneous with respect to data characteristics (*i.e.*, bounded numerical, unbounded numerical, exclusive categorical, and inclusive categorical; shown in color) or inhomogeneous (shown in white). (b) Cluster Bricks were added above and below the arch for the dimension groups.

two differently clustered heat maps are depicted by drawing straight lines between identical data items. Other systems, in which the connectedness of elements is employed as an additional aid, are Semantic Substrates [29], VisLink [4], Caleydo's Bucket [18], Circos [17], and MizBee [22]. The excessive use of connection lines can introduce visual clutter. Bundling of the connections is one way to alleviate this problem. For example, hierarchical edge bundling [13] accounts for structural information about the data to merge the connections hierarchically.

Our own approach, which is detailed in the following section, integrates and advances these ideas into a flexible technique that combines thoughtful arrangement and linking for multiform visualizations of subsetted inhomogeneous data.

4 The VisBricks Approach

For large data sets, it has proven efficient to follow Keim's *Visual Analytics mantra*: "Analyse First, Show the Important, Zoom, Filter and Analyse Further, Details on Demand" [14]. VisBricks embrace this paradigm and strive to support it on all levels by providing meaningful **preprocessing and overviews** to show the important features even for inhomogeneous data; a rich set of interactions to enable **zooming, filtering and further analysis**; and drill down methods to explore even large data sets down to the **details** of the individual record. The core paradigm of VisBricks is to divide-and-conquer: the data set is divided into homogeneous subsets that can then be efficiently abstracted. VisBricks fully support the inhomogeneity of the data and the diversity of tasks at each level of the mantra through their multiform approach, which permits users to tailor the visual representation of each subset of the data according to its characteristics, the task that is to be performed, and the level of detail required.

In this section, we explain the conceptual foundations of the VisBricks technique, beginning with the overview, continuing with interaction aspects that enable zooming and filtering, and finally providing details about how the data is presented on a fundamental level.

4.1 Preprocessing and Overview

Abstraction is a key technique that enables an overview with limited visual or computational resources. There are several ways to achieve abstraction. Oliviera and Levkovitz [9] list dimension reduction, subsetting (*e.g.*, random sampling [5]), aggregation [8], and segmentation (*e.g.*, cluster analysis [7]).

An inherent property of homogeneous data is its suitability for abstraction. With homogeneous data, it is easy to choose a visual encoding that represents the data well. Inhomogeneous data, however, does not lend itself to reasonable abstractions. It is difficult or even impossible to find representative encodings for a very inhomogeneous data set.

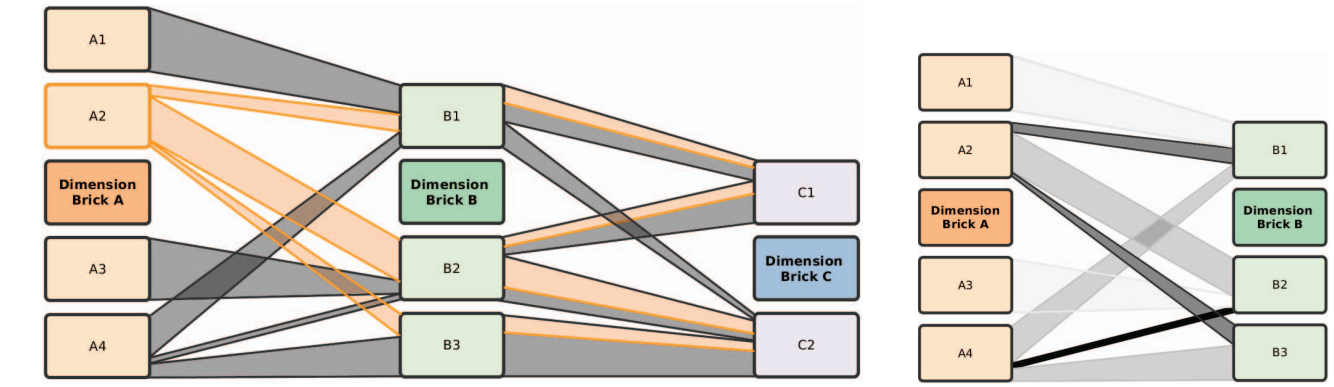
This observation triggered the development of VisBricks. Because VisBricks uphold the divide-and-conquer strategy, they represent subsets of the data that have been generated by vertical and horizontal subdivision that are then aligned vertically and horizontally in dedicated drawing areas. We call the resulting homogeneous groups and their representations *bricks*, as they are the building blocks of the whole visualization and represent their subset of data. These bricks are then placed in the context of the whole data set again: relationships are shown by position and by visual links. This process is achieved in the following four steps:

1. Dividing an inhomogeneous data set into **homogeneous groups of dimensions**.
2. Dividing the records in the homogeneous dimension groups into **homogeneous groups of records**.
3. Placement of the groups of records back into context.
4. Placement of relationships between the dimension groups back into context.

Following the two division steps, we distinguish between two types of bricks: bricks abstracting a whole group of dimensions after the first division, which we call *Dimension Bricks* (as they are representative of the whole dimension group), and bricks reflecting the subdivision of records within the dimension group, which are called *Cluster Bricks* (as the subdivision is often achieved using automatic clustering algorithms). The most important properties of a brick are that it can encode its data in any number of ways and that it lets the user choose the technique to use while providing sensible defaults.

A more detailed look at the four individual steps generating the VisBricks overview layout is given in the following.

Division of Dimensions The actual division of dimensions can be achieved in a number of ways. In many cases, the dimension groups are created manually, because the homogeneity of the dimensions, especially on the semantic level, can often be judged best by users. Alternatively, automatic approaches are possible in an analyse-first step that first divides, for example, the dimensions on a data-type basis, followed by a correspondence analysis to determine similar dimensions. In the interactive case, Dimension Bricks are dynamically added to the dimension arch in VisBricks. Figure 3(a) shows an illustration in which several dimension groups were created and can now be found in the arch. The arch has three regions: the center, where dimension groups that are currently in the focus of the investigation, are placed, and two legs, one on each side, where dimension groups are moved, when they are not in focus. Notice that in the example in Figure 3(a),



(a) Selection propagation (b) Interactive trends filter

Fig. 4. Ribbons connect bricks between adjacent dimension groups, thus indicating how many elements are shared among them. In (a) the user has selected brick A2. The selection is propagated to all connected bricks. (b) shows the result of an interactive filter that focuses on outliers. The wider the connection band is, the lighter it is drawn.

some dimension groups are homogeneous in terms of their dimension characteristics, whereas others are not. They may, however, be homogeneous in terms of semantics.

Division of Records When the data set is divided into dimension groups, the user can choose to further divide the groups horizontally using, for example, clustering algorithms (see Figure 3(b)). Notice that this step is optional, as some dimension groups may not require further subdivision or may not be suitable for it. For other dimension groups, however, only this additional division makes it possible to create meaningful abstractions for the major trends in the data set. As can be seen in Figure 3(b), Cluster Bricks are shown above or below their respective Dimension Bricks, but only for dimension groups in the focus region.

Naturally, the achievable homogeneity of the Cluster Bricks depends on many factors. First, a sensible trade-off has to be found between the number of clusters and their degree of homogeneity. While our VisBricks implementation takes several measures to avoid clipping data, the number of clusters and therefore the number of Cluster Bricks has the greatest impact on the VisBricks' scalability. Second, the achievable degree of homogeneity for a given number of clusters depends on many factors, such as the choice of clustering algorithm, its parametrization, and the suitability of the data set. After this division has been accomplished, we are able to choose suitable visualizations for each brick to abstract the now homogeneous subset of data.

Encoding Relationships between Cluster Bricks When exploring tabular data in a spreadsheet, sorting of the data is a common strategy to find related records. Generally speaking, all visualization techniques that use rows or columns to identify records can make use of sorting. Techniques that encode relationships in a record differently, *e.g.*, parallel coordinates, cannot employ sorting for that purpose.

When sorting by a single row in tabular arrangements, the other values in a record are re-positioned accordingly. Sorting of multiple rows at the same time, however, breaks the ties between the values in the records. Sorting by more than one dimension simultaneously is equally desirable but much harder to achieve, as meaningful comparisons between tuples of values are more difficult to obtain. Consequently, few techniques are able to achieve such sorting. One notable exception is the table-based visualization for bipartite graphs [27], in which the disjoint sets of the graph are visualized in tables and sorting can be performed for each of the sets independently and also simultaneously. Because of the nature of the underlying data (a bipartite graph), no special care has to be taken to keep the association between the records intact.

The Matchmaker technique employs sorting based on averages of clusters [19]. VisBricks adopts this general idea and enhance it by additionally encoding the relationship of every brick to the average of the whole dimension group. The vertical position of a Cluster Brick is determined by two factors: the ranking according to the sorting strategy used and the relative value compared with the average of the whole dimension group. Because of the placement relative to the whole dimension group's average, the Dimension Brick and the arch divide the Bricks into those above the average and those below it. Thus, it is clear how each Cluster Brick compares to the other Cluster Bricks within a dimension group and to the overall average.

Sorting strategies for numerical data would, for example, place the Cluster Brick with the highest average at the top and the brick with the lowest average at the bottom, whereas categorical data could be sorted by frequency. If no meaningful sorting strategy can be defined for a certain type of data, the bricks could be sorted to minimize crossings. For cases where no meaningful average (such as the mean or median) can be defined, the bricks are distributed evenly above and below the Dimension Brick.

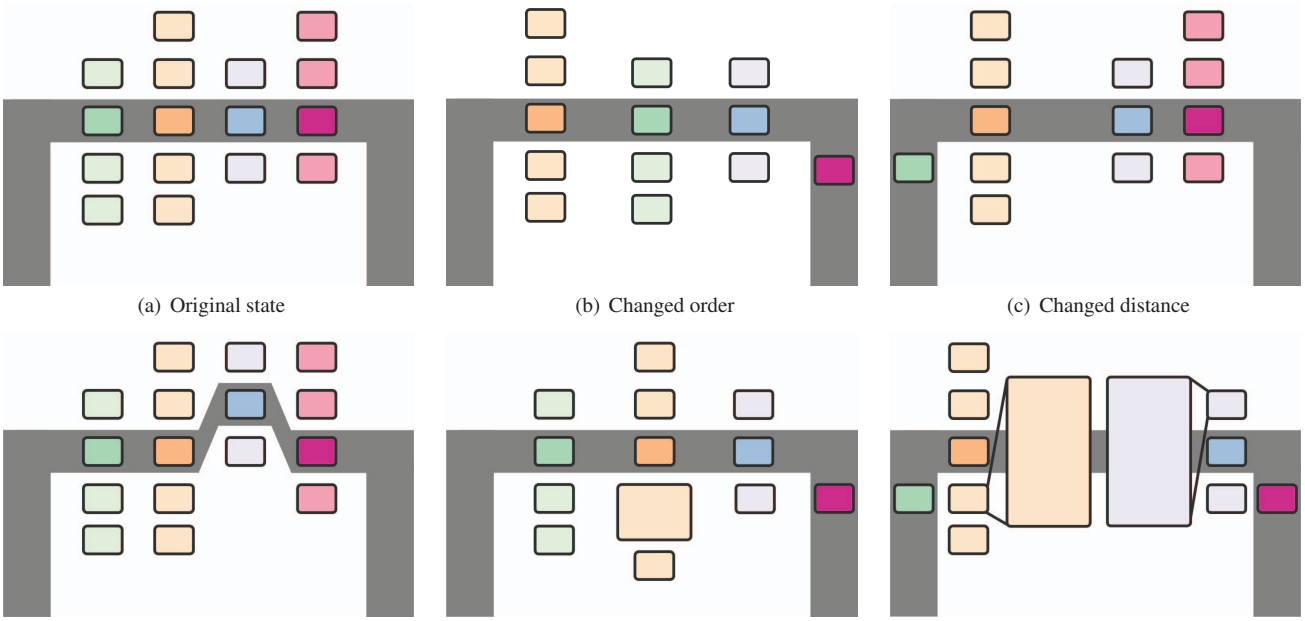
By partitioning and sorting the data records separately in the different dimension groups, the association between individual values of a record across dimension groups is no longer obvious, as the strict horizontal and vertical alignment of the data matrix has been broken up. Hence, the following **conquer** steps reintroduces this essential information in the overall layout of the bricks by encoding the relationships through visual links.

Encoding Relationships between Dimension Groups Finally, to provide a meaningful overview, the relations between the dimension groups must be made explicit, thus realizing the conquer step. We achieve this by using both traditional **linking and brushing** as well as **interactive visual links**.

VisBricks employ ribbons for conveying which portion of the data contained in each brick is shared among bricks in neighboring dimension groups. When the bricks are brushed, the ribbons are not only shown for the relationships to the neighboring dimension groups but also split into multiple threads connecting all related bricks in all dimension groups (see Figure 4(a)). The width of the ribbons encodes the magnitude of the relation. The ribbons are sorted to minimize crossings. This strategy is similar to the one used in Parallel Sets [16].

It is possible to brush only the ribbon connecting two bricks, thereby focusing on the subset of data shared by those two bricks. The brushing of bricks or ribbons can also be reflected in the views contained in the bricks. VisBricks support multiple simultaneous brushes, assigning a different color to each brush. In cases with many clusters, it is sensible to show ribbons only for brushed bricks.

Whereas wide ribbons show major trends among the dimension groups, thin ribbons indicate outliers. Initially, the showing of both outliers and major trends is a good option to convey an overview. However, in many tasks either only outliers or only major trends are relevant. We therefore developed a technique that allows users to in-



(a) Original state (b) Changed order (c) Changed distance

(d) Changed vertical position (e) Changed brick size (f) Focus duplicates

Fig. 5. Interaction patterns in VisBricks. (a) shows the original state, whereas (b)-(f) show the consequences of the five different interaction patterns.

teractively specify whether they are currently interested in the main trends, outliers, or anything in between. Because "outlier" or "main trend" are not absolute concepts, we chose to decrease the opacity for bands further from the current focus. Figure 4(b) shows an example in which the focus lies on outliers.

4.2 Zoom, Filter and Analyze Further

The provision of overviews is essential in making it possible to understand a data set. However, to extract knowledge, it is necessary to drill down, either via interactive zooming and filtering or via a reparameterization of the analysis, *e.g.*, by refining the clusters. While the latter is not a matter of the visualization itself, the interactive zooming and filtering are performed directly on the visualization and should thus be supported by it. VisBricks provide this support through five interaction patterns for manipulating the bricks and their layout.

1. **Changing the order of dimension groups**
   Dimension groups can be moved in and out of the focus region; the latter provides more space for those in focus. Additionally, the horizontal order can be rearranged, allowing a more detailed side-by-side comparison of different dimension groups. When dimension groups are brought into focus manually, others can be forced out of focus and into the context region if more space is required than available. Figure 5(b) illustrates an example in which the order of dimension groups in the focus region was changed and one dimension group was moved to the right leg.
2. **Changing the distance between dimension groups**
   It can be desirable to change the spacing between dimension groups. Increased space is useful if the relationships between two neighboring dimension groups are under investigation. In this case, the increased space reduces the clutter produced by the ribbons. A reduction of space is typically achieved automatically when the space is increased elsewhere. In Figure 5(c), the orange dimension group was moved to the left, which pushed the green dimension group out of focus into the leg.
3. **Changing the vertical position of dimension groups**
   By changing the vertical position of the dimension groups, Cluster Bricks, which are close to or even beyond the border of the screen, can be moved into the center, and comparisons between two bricks of neighboring dimension groups are facilitated. As shown in Figure 5(d), the arch is bent, if necessary, to guarantee that it always encloses the Dimension Brick.
4. **Changing the size of a brick**
   Each brick can be resized so that the containing visualization receives more space, as shown in Figure 5(e). When the space for a brick is increased, other bricks are moved upwards or downwards, and other dimension groups are moved to the side. Again, dimension groups are moved to the legs if necessary.
5. **Creating a focus duplicate of a brick**
   When a full-sized visualization is more suitable for a given task, VisBricks provide the means to allow a brick to temporarily claim additional space for an enlarged focus mode. However, this focus mode is not simply an enlarged version of a brick, which would be achievable using only the resize functionality. Instead, the focus mode provides means (a) to compare single bricks in detail to another dimension group, (b) to compare this brick in detail to a second brick of another dimension group, and (c) to prevent the other bricks of the same dimension group from being clipped. The focus mode is chosen for a single brick of interest, which is then duplicated and placed next to its dimension group. By choosing the side of the dimension group on which the brick is to appear, the target of the comparison is implied. When the detail brick is visible, its connections to the neighboring dimension group appear. A user can now analyze the relationships and choose a brick from the compared dimensions for detailed analysis. Figure 5(f) illustrates the state in which a second brick is enlarged. For some visualization techniques, the available horizontal space may not be sufficient. In such cases, the legs of the brick are moved out of the view, to increase the space for the focus bricks.

Especially with this last interaction technique, it becomes apparent that a drill-down from the overview, which only shows the important data in abstracted views, to detailed views of individual homogeneous subsets is fully supported by VisBricks. Additional consideration regarding the detailed visual analysis of individual data properties is discussed in the following section.

4.3 Exploring Details

The detailed analysis in VisBricks is based on the multiform property of the bricks. Although we previously mentioned that multiple visual-

ization techniques can be used within a brick, we have up to this point mainly treated bricks as a medium to present abstractions. However, the bricks are more powerful.

The defining property of the bricks is their ability to display the information grouped within them using diverse visualization techniques. We have distinguished between Dimension Bricks, which summarize the entire data in a dimension group, and Cluster Bricks, which show data that is homogeneous in terms of statistics. Both require very different visualizations, as the Dimension Bricks give an overview of the grouped dimensions, whereas the Cluster Bricks show the records grouped inside them. In general, it is not immediately obvious which visualization is sensible for which brick. The suitability of a technique depends on two criteria:

1. **Data characteristics criterion:** Is it suitable to visualize the data for the given data characteristics?
2. **Scalability criterion:** Is it suitable to visualize the given amount of data in the allocated space?

Data Characteristics Criterion For bricks that are homogeneous with respect to their data characteristics, it is easy to assign suitable visualizations. The availability of a concrete visualization technique as a representation choice for such a brick requires only the identification of the data characteristics for which it is suitable. An example would be a parallel coordinates view, which is suitable for bounded numerical, unbounded numerical, and, to some extent, exclusive categorical but not for inclusive categorical.

However, when dimension groups are not homogeneous with respect to their characteristics but only with respect to their semantics, it is not as simple to assign suitable visualizations. In this case, we seek out the "least common representation" that is sufficiently generic to be able to show all of the data types within such a mixed dimension group. To achieve this, we order the data types according to their *strictness* for the data characteristics. For the four data types, we consider bound numerical to be the strictest characteristic, followed by unbound numerical, exclusive categorical, and, finally, inclusive categorical as the most *relaxed* type. This ordering is based on the observation that data belonging to a stricter class can often also be visualized with a technique suitable for a more relaxed data type. What distinguishes visualization techniques for stricter classes from those for more relaxed classes are the assumptions about certain properties of the data that do not hold for more relaxed types. An example would be a technique for bounded numerical values that assigns each record a hue of 1 for the upper bound and 0 for the lower bound. If this technique is used with a hybrid dimension group, in which one dimension contains unbound values, their color coding becomes meaningless.

Visualization techniques for more relaxed data types have to allow their records to take on a wider variety of states, making the individual record more expressive, but also harder to abstract. This does not mean that a technique for a more relaxed characteristic is not suitable for a stricter characteristic; rather, it means, that such a judgment cannot be derived automatically.

Note that it is not reasonable to employ a technique that is suitable for more relaxed characteristics to all stricter ones. Usually, more relaxed techniques are not able to fulfill the scalability criterion as well as stricter techniques do.

Scalability Criterion VisBricks rely heavily on the abstraction technique of segmentation into homogeneous groups at the top level, and in fact we employ a multi-level approach: bricks are required to provide at least one abstraction method for every data characteristic. Hence, each visualization technique can make use of the provided abstraction methods as needed. Dix and Ellis note that multi-level abstraction solutions are common; for example, a sampled data set can be used as the input for aggregation techniques [5].

We distinguish among four classes of bricks, where each has different requirements considering the scalability criterion:

1. **Regular Dimension Bricks**
   Dimension bricks represent all records in a dimension group. As the number of records can be large, techniques that rely on the scaling of width or height with the number of records are not suitable for Dimension Bricks. Consequently, aggregation methods, such as histograms, or methods using sub-setting and natural aggregation, such as parallel coordinates, are suitable. In contrast, methods that require additional space for every record, *e.g.*, clustered heat maps [7, 31] or tables, are not suitable. A Regular Dimension Brick is shown in Figure 6(a).
2. **Compact Dimension Bricks**
   When a dimension group is moved to the legs of the arch, its Cluster Bricks are hidden, and the Dimension Brick is reduced to a static size, optionally showing a high-level aggregation of the data. Figure 6(b) shows an example for numerical data, in which the whole dimension group is aggregated into one single line of a heat map. Although this abstraction is very crude, it may show a major trend in the data.
3. **Regular Cluster Bricks**
   Regular Cluster Bricks have the most freedom of all bricks. They may use any visualization technique suitable for the data, including those that require the scaling of width and height with the number of records. For example, Figure 6(c) shows a Cluster Brick containing a parallel coordinates view. However, basically any imaginable visualization technique that is able to provide an overview of a number of multi-dimensional records can be used inside a Regular Cluster Brick.
4. **Compact Cluster Bricks**
   For each data characteristic, VisBricks require one technique that represents a cluster at minimal height. This technique is used by default if the bricks would otherwise not fit in the view. Although this technique cannot completely avoid clipping, it significantly increases scalability. The actual height is not specified, because, for example, efficient visual abstractions of categorical data are much more difficult to achieve than those for numerical data. Compact Cluster Bricks have a reduced set of user interface elements, which help to keep the size minimal. Figure 6(d) shows an example for numerical data, in which a heat map line, similar to the abstraction used in the Compact Dimension Brick, shows an aggregation of the cluster. Under the assumption that the records in the brick are in fact homogeneous, this abstraction is a valid representation for the cluster.

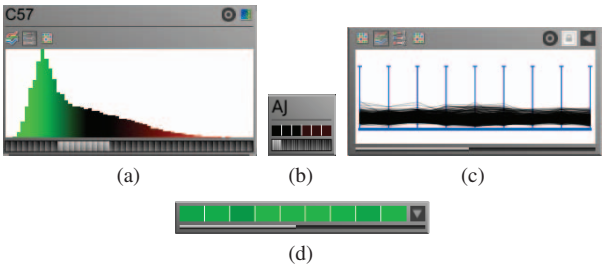


Fig. 6. The different classes of bricks used in VisBricks. (a) Regular Dimension Brick, summarizing a dimension group. (b) Compact Dimension Brick, used in the arch legs. (c) Regular Cluster Brick, showing one homogeneous cluster. (d) Compact Cluster Brick used for overviews.

In addition to these four fundamental modes, views are also notified of the actual size of a brick to enable them to adapt to the available resolution, which makes it possible to prevent users from switching to visualization techniques that require more space than the brick currently has available or to adapt the level of detail. The parallel coordinates, for example, add captions when a certain size threshold is surpassed and user interface elements when the view is enlarged further.

With these scalable bricks at hand, users can interact with the data, drill down into clusters of dimension groups, explore the details of relationships between clusters and dimension groups, and even see the actual values of every single record in the data. In Section 6, we will present the results achievable with a prototype implementation. How-

ever, we will first discuss some design choices, implementation details and scalability issues.

5 Practical Considerations

5.1 Design Choices

In addition to the main paradigms already discussed in the previous section, there are some additional considerations to improve the usability of bricks.

One piece of information that is lost when abstracting homogeneous groups for dimensions and records is the scale of the group. A homogeneous brick containing only a few elements would, for example, be assigned the same space as another brick containing half the data set. It is therefore necessary to encode the relative size of the groups in terms of the number of dimensions for the dimension group and the number of records for the Cluster Bricks. To encode the number of dimensions, we use a row of squares with one square for each dimension; the squares are filled if this dimension is part of the dimension group (see Figures 6(a) and (b)). We encode the number of records in the Cluster Bricks with a bar, as shown in Figures 6(c) and (d).

Also, the bricks need to contain user interface elements to, for example, display the name of a dimension group or allow switching between visualization techniques. Many approaches are conceivable. For our prototype, we chose a mixture between static and pop-up buttons, which can be seen in Figure 6.

5.2 Implementation

We realized the VisBricks technique as a prototype in the Caleydo information visualization framework[2][18]. For numerical data, we provide a parallel coordinates implementation, a heat map, a histogram, and the required abstract views. For clustering, we use Affinity Propagation [10] and the Weka implementation of k-means [11].

Except for the red-green color mapping commonly used in biomedicine, all color-schemes, for the figures in this paper and the application, are taken from Colorbrewer [3]. To accommodate red-green blind users, Caleydo provides alternative color schemes.

The VisBricks technique is implemented in OpenGL using the Java OpenGL Binding[3]. We use the Eclipse RCP framework and plug-in mechanism. Through this mechanism, views for VisBricks can be added without access to the source code. The layout of all elements is recursively defined with a specially designed, flexible layout package.

5.3 Scalability

VisBricks scale to a large number of records and dimensions. The primary limiting factor for the number of records is the computational limitation of the clustering algorithms. A secondary limitation is the available resolution: On a WSXGA+ screen with 1680×1050 pixels, VisBricks can handle up to 30 clusters in one dimension group. The cluttering of connections associated with a high number of clusters between many dimension groups can be improved by rendering ribbons only when brushed, or by using the trend filter. VisBricks can accommodate about ten to fifteen dimension groups, up to eight of which may be in the focus region.

6 Usage Scenario

We evaluated our tool with a data set from our partners at the Institute of Pathology at the Medical University of Graz. Their team focuses on determining the genetic factors of liver cirrhosis. Cirrhosis is a multifactorial disease, which means that many factors, both environmental and genetic, influence its development. Contrary to popular perception, cirrhosis is linked not only with alcoholism, but also with diabetes and obesity. Our partners have developed a mouse model that allows them to monitor the progression of steatohepatitis (fatty liver disease), which is a disease that eventually leads to cirrhosis. They perform experiments in which steatohepatitis is induced by feeding mice poison for eight weeks. They discovered that different genotypes (genetic types) of mice are not equally prone to develop symptoms. To uncover the genetic factors that lead to this discrepancy, they measure the gene expression of the mice at different times.

We have previously worked with this data set [19] to demonstrate differences between clustering algorithms and to exemplify an analysis using multi-level heat maps. The VisBricks concept goes well beyond the previously presented approach in supporting the full range of visual analysis, from a comprehensive overview of the topology of the entire data set that integrates diverse computational and visual options, seamlessly down to the individual data record.

Following the Visual Analytics mantra, the computational analysis constitutes the first step. In this data set, there are multiple levels of semantic inhomogeneities, *i.e.*, experiments conducted at different points in time or with different genotypes of mice. Sensible groupings of the data depend on the research question. Thus, if changes over time comprise the main focus, grouping based on similar points in time would be the best choice. However, because the differences in genotype are central to the research question, grouping based on genotypes makes the most sense. To avoid insignificance within control groups, the analyst filters the data using statistical methods and also removes values that are constant within a threshold across all conditions. The filtered data set shown in Figure 7 has 37 dimensions, each containing the measurements of 1 experiment, grouped by the 7 different genotypes of mice, with 766 expression values per dimension.

In this scenario, the analyst is interested in differences between the AJ genotype and the other genotypes (C* and PWD) because mice with the AJ genotype are less susceptible to steatohepatitis. An example of a relevant difference is gene expression is a gene that remains at the same regulatory level in the AJ mice but is upregulated as time progresses in the C* mice. Such a gene might be involved in causing steatohepatitis in the C* mice.

Figure 7(a) shows the layout of the Dimension Bricks, one for each of the seven genotypes as an overview of the data set. Two dimension groups have already been clustered, and their corresponding Cluster Bricks are shown. The histograms in the Dimension Bricks show the summarized distribution of the values in the dimension groups, from low expression (at the left in green) to over-expression at the right in red. Subtle differences between the dimension groups are noticeable.

The analyst then proceeds by clustering the remaining dimension groups to uncover their statistical inhomogeneities. As the dimensions within the dimension groups are sorted by time (early experiments are on the left, whereas the final measurements are on the right), there is a visibly strong tendency of increased expression from left to right in the appearing Cluster Bricks in all dimension groups.

The clustering groups together those genes with similar expression patterns. Such groups are often also functionally similar [7], making the clusters semantically meaningful. In looking for differences between a gene's expression in the AJ and the C* mice, the analyst is searching for two clusters that share elements (*i.e.*, they are connected with ribbons) but also show a different behavior for the genotypes. As the mice are treated exactly the same, such a difference is likely to stem from the difference in genotype and might thus be linked to the causes of steatohepatitis.

The analyst begins a more detailed analysis by filtering. He moves some dimension groups to the arch legs to take a close look at the differences of the dimension groups of interest. To see some of the more interesting bricks in detail, the analyst switches them to the parallel coordinates view. Other, less interesting Cluster Bricks, in which values remain nearly constant over time, are switched to the compact mode. The many broad ribbons between closely related Cluster Bricks show that much of the data is largely consistent across the dimension groups, indicating that those genes behave similarly in the different genotypes. However, there are connections between rather distant Cluster Bricks, hinting at possible outliers. Using interactive colored brushing, the analyst explores the relationships of selected Cluster Bricks in more detail. The brushing highlights the ribbons and the actual data in the parallel coordinates. When brushing the Cluster Brick that shows the parallel coordinates in the second dimension group (orange brushing in Figure 7(b)), the analyst notices one brick in the neighboring dimension group that is far away and very dissimilar. However, it still

[2] http://caleydo.org
[3] http://jogamp.org

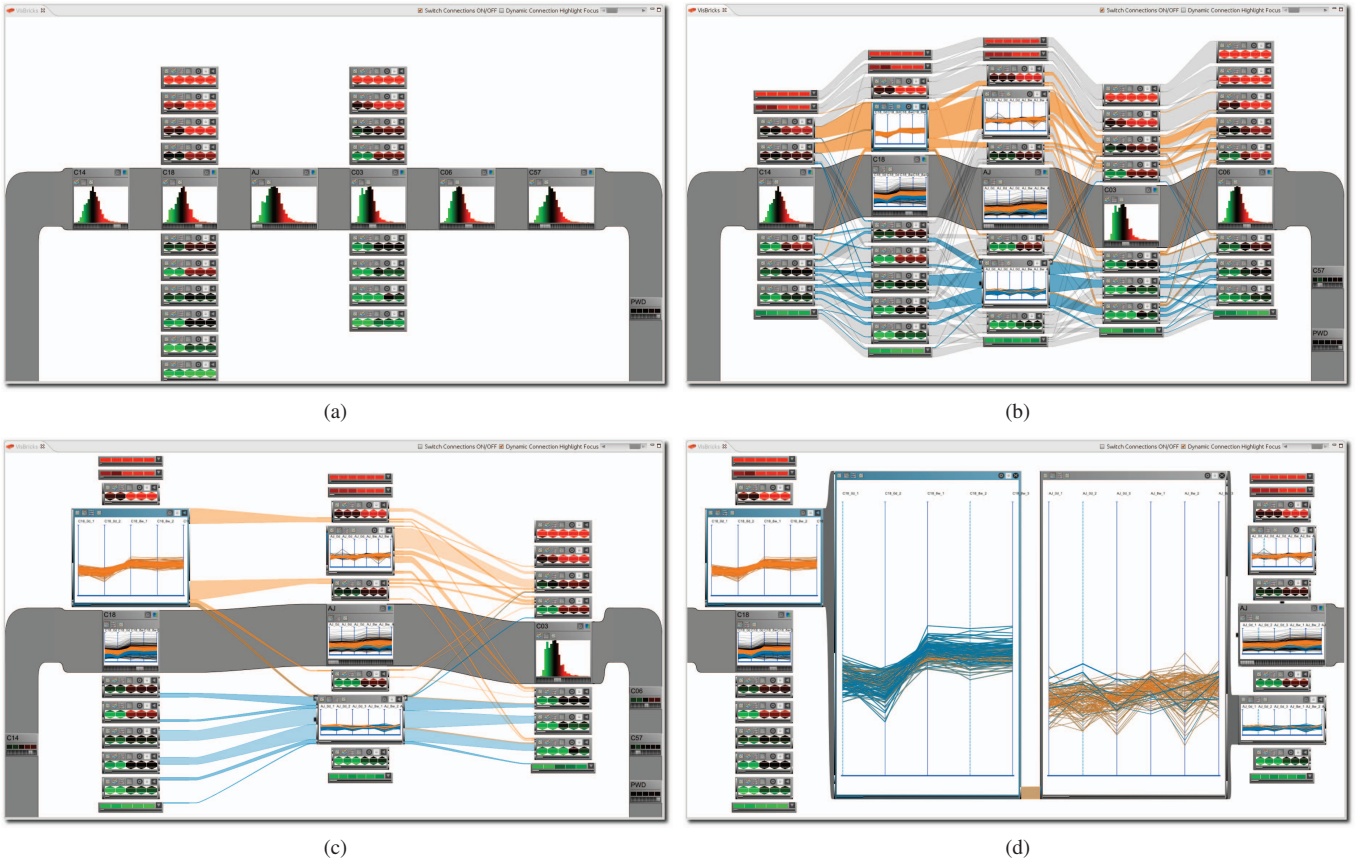


Fig. 7. Steps in an analysis of gene expression in different genotypes of mice. (a) The overview with two dimension groups clustered. (b) All dimension groups clustered and two bricks brushed. Notice the connection from the top-left brick showing the parallel coordinates (orange brush) to the lower brick in the center (blue brush). The blue brick contains outliers of the orange brick, which indicates genes of interest. (c) Enlarged bricks of interest, showing only ribbons for the outliers of brushed bricks. (d) The bricks of interest in focus mode, which enables detailed analysis.

shares a few records with the brushed brick. The analyst switches the brick's aggregative view containing the outliers to a parallel coordinates view, where the outliers are immediately obvious. To explore the outliers in more detail, the analyst increases the size of the bricks and chooses to show only ribbons for outliers of brushed bricks (see Figure 7(c)). The shared records seem interesting and deserve closer investigation. The analyst therefore switches to the focus mode, as shown in Figure 7(d), where the genes are explored in detail using two parallel coordinates views. The genes found may indeed play a role in steatohepatitis. The analyst continues to investigate by reviewing the literature on the found genes in online databases.

Note that the accompanying video shows this usage scenario, including all intermediate steps not discussed in this text.

The feedback from our partners was very positive; they were able to conduct an analysis only after a brief training period, in which the novel spatial arrangement and the meaning of the ribbons were explained. Our partners appreciated the interactivity of the system and its ability to focus on several different parts of the data at the same time. They noted that this was very hard to achieve in their previous workflow using earlier versions of Caleydo, other state-of-the-art microarray analysis tools, or statically generated R-plots. An interesting suggestion made was to integrate other, non-tabular data sources, such as pathways, into VisBricks as well.

7 Conclusion and Future Work

We have shown that the VisBricks concept can handle large and inhomogeneous data spaces by employing it in a real-life, complex analysis scenario. The main advantage of VisBricks, compared with traditional approaches is their ability to handle all types of inhomogeneities within data, both in the dimensions and in the records. This is achieved by treating each homogeneous sub-part of the data with the best available computational and visual tools. By using abstractions in the bricks, VisBricks are very scalable in terms of the magnitude of records. At the same time, the division into bricks and the rich set of interaction patterns allow users to employ multi-level approaches, in which each brick contains an abstraction suitable to show the data at the desired level of detail.

The VisBricks concept is sufficiently powerful to describe previous visualization approaches in terms of bricks, groups, and the relationships among them. One example for categorical data is Parallel Sets [16]. Each brick can represent a category, and Parallel Sets' "composed dimensions" can be interpreted as dimension groups. Parallel Sets optionally show histograms inside the categories, which is also possible in bricks. For numerical data, Matchmaker [19] can be formulated in terms of bricks. The clusters shown in the heat maps used in Matchmaker are essentially small bricks.

At the very core of the VisBricks strategy are two concepts: the creation of homogeneous sub-parts of the data and the establishment of multiform visualization for those parts. These concepts are in no way limited to tabular data; they may also be applied to other data forms. However, the encoding of topological structures, positions, and connections between more general forms of data, such as geo-spatial data, will be the subject of future research.

Acknowledgments

Thanks to our partners at the Medical University of Graz. This work was funded in part by the Austrian Research Promotion Agency (FFG) through the *inGeneious* project (385567) and the *Caleydoplex* project (P22902) granted by the Austrian Science Fund (FWF).

REFERENCES

[1] J. Abello, F. Ham, and N. Krishnan. ASK-GraphView: a large scale graph visualization system. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):669–676, 2006.

[2] D. Archambault, T. Munzner, and D. Auber. TopoLayout: multilevel graph layout by topological features. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):305–317, 2007.

[3] C. A. Brewer. Colorbrewer. http://colorbrewer2.org/, last accessed March 30, 2011, 2009.

[4] C. Collins and S. Carpendale. VisLink: revealing relationships amongst visualizations. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '07)*, 13(6):1192–1199, 2007.

[5] A. Dix and G. Ellis. By chance: enhancing interaction with large data sets through statistical sampling. In *Proceedings of the ACM Conference on Advanced Visual Interfaces (AVI '02)*, pages 167–176. ACM Press, 2002.

[6] C. Eaton, C. Plaisant, and T. Drizd. Visualizing missing data: Graph interpretation user study. In *Proceedings of the Conference on Human-Computer Interaction (INTERACT '05)*, volume 3585 of *Lecture Notes in Computer Science (LNCS)*, pages 861–872. Springer, 2005.

[7] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences USA*, 95(25):14863–14868, 1998.

[8] N. Elmqvist and J. Fekete. Hierarchical aggregation for information visualization: Overview, techniques, and design guidelines. *IEEE Transactions on Visualization and Computer Graphics*, 16(3):439–454, 2010.

[9] M. Ferreira de Oliveira and H. Levkowitz. From visual data exploration to visual data mining: a survey. *IEEE Transactions on Visualization and Computer Graphics*, 9(3):378–394, 2003.

[10] B. J. J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315(5814):972–976, 2007.

[11] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: an update. *SIGKDD Explorations*, 11(1):10–18, 2009.

[12] N. Henry, J. D. Fekete, and M. J. McGuffin. NodeTrix: a hybrid visualization of social networks. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '07)*, 13(6):1302–1309, 2007.

[13] D. Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '06)*, 12(5):741–748, 2006.

[14] D. A. Keim, F. Mansmann, J. Schneidewind, and H. Ziegler. Challenges in visual data analysis. In *Proceedings of the Conference on Information Visualisation (IV '06)*, pages 9–14, 2006.

[15] W. Klsgen. Types and forms of data. In W. Klsgen and J. M. Zytkow, editors, *Handbook of data mining and knowledge discovery*, pages 33–44. Oxford University Press, 2002.

[16] R. Kosara, F. Bendix, and H. Hauser. Parallel sets: Interactive exploration and visual analysis of categorical data. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):558–568, 2006.

[17] M. Krzywinski, J. Schein, I. Birol, J. Connors, R. Gascoyne, D. Horsman, S. J. Jones, and M. A. Marra. Circos: An information aesthetic for comparative genomics. *Genome Research*, 19(9):1639–1645, 2009.

[18] A. Lex, M. Streit, E. Kruijff, and D. Schmalstieg. Caleydo: Design and evaluation of a visual analysis framework for gene expression data in its biological context. In *Proceeding of the IEEE Symposium on Pacific Visualization (PacificVis '10)*, pages 57–64. IEEE Computer Society Press, 2010.

[19] A. Lex, M. Streit, C. Partl, K. Kashofer, and D. Schmalstieg. Comparative analysis of multidimensional, quantitative data. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '10)*, 16(6):1027–1035, 2010.

[20] A. MacEachren, D. Xiping, F. Hardisty, D. Guo, and G. Lengerich. Exploring high-D spaces with multiform matrices and small multiples. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis '03)*, pages 31–38. IEEE Computer Society Press, 2003.

[21] S. Mansmann and M. H. Scholl. Exploring OLAP aggregates with hierarchical visualization techniques. In *Proceedings of the ACM Symposium on Applied Computing (SAC '07)*, pages 1067–1073. ACM Press, 2007.

[22] M. Meyer, T. Munzner, and H. Pfister. MizBee: a multiscale synteny browser. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '09)*, 15(6):897–904, 2009.

[23] T. Nocke and H. Schumann. Meta data for visual data mining. In *Proceedings of the Conference on Computer Graphics and Imaging (CGIM '02)*, 2002.

[24] T. Pham, R. Hess, C. Ju, E. Zhang, and R. Metoyer. Visualization of diversity in large multivariate data sets. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '10)*, 16(6):1053–1062, 2010.

[25] J. C. Roberts. Multiple-View and multiform visualization. In *Visual Data Exploration and Analysis VII, Proceedings of SPIE*, volume 3960, pages 176–185, 2000.

[26] S. Rufiange, M. J. McGuffin, and C. Fuhrman. Visualisation hybride des liens hirarchiques incorporant des treemaps dans une matrice d'adjacence. In *Proceedings of the Conference on Association Francophone d'Interaction Homme-Machine (IHM '09)*, pages 51–54, 2009.

[27] H. Schulz, M. John, A. Unger, and H. Schumann. Visual analysis of bipartite biological networks. In *Proceedings of the Eurographics Workshop on Visual Computing for Biomedicine (VCBM '08)*, pages 135–142. Eurographics, 2008.

[28] J. Seo and B. Shneiderman. Interactively exploring hierarchical clustering results. *Computer*, 35(7):80–86, 2002.

[29] B. Shneiderman and A. Aris. Network visualization by semantic substrates. *IEEE Transactions on Visualization and Computer Graphics (InfoVis '06)*, 12(5):733–740, 2006.

[30] J. J. Thomas and K. A. Cook. *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. National Visualization and Analytics Center, 2005.

[31] L. Wilkinson. The history of the cluster heat map. *The American Statistician*, 63(2):179–184, 2009.

[32] A. Woodruff, C. Olston, A. Aiken, M. Chu, V. Ercegovac, M. Lin, M. Spalding, and M. Stonebraker. DataSplash: a direct manipulation environment for programming semantic zoom visualizations of tabular data. *Journal of Visual Languages & Computing*, 12(5):551–571, 2001.

[33] J. Yang and S. Barlowe. A dimension management framework for high dimensional visualization. In *Advances in Information and Intelligent Systems*, number 251 in Studies in Computational Intelligence, pages 267–288. Springer, 2009.

[34] S. Zhao, M. McGuffin, and M. Chignell. Elastic hierarchies: combining treemaps and node-link diagrams. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis '05)*, pages 57–64. IEEE Computer Society Press, 2005.