NISS

Visual Scalability

Stephen G. Eick and Alan F. Karr

Technical Report Number 106
June, 2000

National Institute of Statistical Sciences
19 T. W. Alexander Drive
PO Box 14006
Research Triangle Park, NC 27709-4006
www.niss.org

Visual Scalability

Stephen G. Eick
Visual Insights, Inc.
215 Shuman Boulevard, Suite 200
Naperville, IL 60563–8495
eick@visualinsights.com

Alan F. Karr*
National Institute of Statistical Sciences
P.O. Box 14006
Research Triangle Park, NC 27709–4006
karr@niss.org

Revision: June 6, 2000

Abstract

Visual scalability is the capability of visualization tools effectively to display large data sets, in terms of either the number or the dimension of individual data elements. In this paper, we define and structure the problem of visual scalability, with special emphasis on the role of visualization as a means of access to details of the data. This is done abstractly in terms of responses that measure the business or scientific impact of visualizations and factors that affect the responses, and concretely in terms of measures of visual scalability and factors influencing them. We assess both current capabilities and future prospects along a number of dimensions. Our approach for increasing visual scalability includes improved visual metaphors, interactivity and perspectives that link multiple views.

*Research supported in part by NSF grants SBE–9529926 and EIA–9876619 to the National Institute of Statistical Sciences.

Contents

List of Figures

1 Introduction

With the widespread instrumentation of business and government, it has now become technically feasible and cost–effective to collect, store and analyze huge volumes of fine-grained transaction data. The availability of large data sets and business need to understand these data has stimulated widespread research on analysis of massive data sets.

Much of this research has focused on development, implementation and analysis of algorithms. These algorithms range from adaptations or extensions of existing techniques for classification and clustering to new methods for data mining. In this setting, much attention has been devoted to scalability of algorithms. Computational complexity is perhaps the largest, but not the only, thread of this research.

Visualization has played an essential role in dealing with large data sets. Tools include both the application-oriented, for example, directed to software characteristics [17] and software changes [16], and the generic, such as DataDesk [34], xGobi [32], NicheWorks [37], XmdvTool [35], Spotfire [2] and ADVIZOR™ [22].

However, while the need is clear, scalability analyses of visualizations are almost entirely absent. Many visualization metaphors, or particular implementations of them, do not scale effectively, even for moderately–sized data sets that can easily be manipulated on current PCs. For example, many bar charts cannot display more than a few dozen bars. Scatterplots, one of the most useful graphical techniques for understanding relationships between two variables, can be overwhelmed by a few thousand points. Visualizations of networks may be vitiated by link overplotting.

Our goals in this paper are to (1) Define and structure the problem of *visual scalability*; (2) Assess current capabilities along a number of dimensions, as well as assess future prospects; and (3) Describe ways to increase visual scalability— focusing on visual metaphors, interactivity, and combinations of individual views into perspectives that exploit the complementary strengths of their components.

Historically, the role of visualization has usually been to facilitate discovery and understanding of high–level structure of the data in ways impossible by direct examination of the data themselves. Increasingly, however, visualizations must also (or even instead) serve as effective interfaces to access details and refined statistical features of the data. This is particularly true in settings such as E-commerce, one of the most significant new forms to data to appear in recent years, one-to-one marketing, and other CRM-related data sets. Much of our presentation is framed in terms of this second role for visualization; as a consequence, some techniques (for example, density estimates as alternatives to scatterplots, which reveal structure but inhibit access to details) receive comparatively abbreviated treatment.

The paper is organized as follows. In §2, we describe the structures in which the data to be visualized are stored and introduce both measures of visual scalability and factors that affect visual scalability. In §3, we assess current and prospective limitations on the factors. §4 describes a number of strategies that we have applied to increase visual scalability, whose effects are not only illustrated with particular data sets but also assessed quantitatively.

2 What is Visual Scalability?

Ideally, visual scalability should be quantified in terms of (1) *Responses* that measure the number and impact of the insights, discoveries and business decisions that a visualization induces; and (2) *Factors* — characteristics of visualizations that affect the responses. Then, models could be developed and validated that quantify dependence of responses on factors and data sets:

$$\text{responses} = F(\text{factors}, \text{data}). \tag{1}$$

Currently, this is not possible, primarily because few responses can be quantified or measured.

As an initial step, we adopt a more modest approach. Responses are replaced by measures of visual scalability formulated in §2.2. Factors affecting visual scalability are introduced in §2.3. No models of the form (1) are developed, although in §3 and 4 we discuss the *direction* of the relationships.

2.1 Data Structures

Consistent with the role of visualization as a means of access to data, we consider visualization of (observational) data stored in databases.[1] Relational databases organize microdata (individual cases) into one or more *tables*, in which rows are records and columns are attributes of the records. For efficiency reasons, large data sets are frequently stored in many related tables with well-known substructure. The relationships among the tables are determined by the database schema. Examples are Census data (records are households, attributes are characteristics such as number of people, employment and income) and E-commerce transaction data (records are sales, attributes include item, quantity and customer). Access to relational databases is typically by means of SQL.

Relational databases can be arbitrarily large. Those containing hundreds of thousands to millions of rows and tens to hundreds of columns can be stored and queried on PCs.

2.2 Measuring Visual Scalability

We divide measures of visual scalability into two classes:

Database metrics that measure the *size of the database*. Examples for relational tables are bytes, number of rows and number of attributes.[2] For data cubes,[3] measures include not only the characteristics of the underlying microdata but also the number of dimensions and hierarchies in the cube.

Visualization characteristics that measure the *number of distinct items* — a combination of data elements and attributes — displayed on the screen. Examples are the number of bars in a bar chart, the number of nodes or links in a network diagram and the number of points in a scatterplot. (Once more, we remind that this point of view is consistent with visualization as access to details of the data.)

[1] Other important contexts include output of computer models — often the domain of *scientific visualization*.

[2] When joins of multiple tables in a relational database are required, the number of tables is another measure of size.

[3] Multi-dimensional databases such as Microsoft SQL Server that aggregate microdata into categories.

2.3 Factors Affecting Visual Scalability

Here we list factors affecting visual scalability, ordered "from the person down."

Human perception. The human eye connects the two most powerful information processing systems — the human mind and the modern computer. For visual interfaces, the precision of the eye and the ability of the human mind to process visual patterns limit the capacity of this connection.

Monitor resolution affects on visual scalability through both physical size of displays and pixel resolution.

Visual metaphors are the means by which data characteristics are encoded for visual display. This involves not only selection of a basic metaphor, such as a bar chart, but also mapping of data attributes onto visual characteristics of the chosen metaphor, such as bar size and color.

Interactivity leverages visual scalability significantly, by mechanisms ranging from traditional pan and zoom to coupled, multi-resolution metaphors, and is discussed in §4.1.

Data structures and algorithms must support visualization needs. For a visualization to scale the data structures and algorithms on which it relies must also scale. Both direct computations needed to produce the visualization itself (for example, graph layout algorithms) and indirect computations (e.g., aggregation of a data cube to fewer dimensions) have strong effects.

Computational infrastructure is primarily an issue of speed of processing (CPU), graphics rendering (graphics card) and data access (via network and hard disk).

In §4 we take the position that new metaphors, interactivity and perspectives that link multiple metaphors offer significant opportunity to increase visual scalability, while many other factors are approaching inherent limits.

3 Assessing Current and Potential Capabilities

For each of the factors described in §2.3, we now assess both its current and future roles in limiting visual scalability. Scalability is also strongly influenced by database characteristics (for reasons ranging from sheer scale to the fact that data lacking an inherent visual metaphor are harder to visualize), but we do not emphasize this dependence here.

3.1 Human Perception

Primarily, limitations of human visual perception affect the visualization characteristics in §2.3. At a normal monitor viewing distance, calculations in [36] suggest that approximately 6.5 million pixels might be perceivable, given sufficient monitor resolution (discussed below).

For visualization as access to details of the data, the number of perceivable pixels affects visual scalability directly. For visualization as a means of discovering structure, the issue, of course, is not whether 6.5 million data points can be displayed, but whether structure and information in the data can be displayed.[4] From either perspective, monitor resolution rather than human vision is currently the limiting factor.

[4] Provocative questions of whether visualizations *distort* characteristics of the data are discussed in [18].

Computer	Typical Resolution	Pixels Displayed
Ultralight	640×480	307,200
Laptop	800×600	480,000
Portable PC	1024×768	786,432
Desktop PC	1280×1024	1,310,720
Graphics Workstation	1600×1200	1,920,000

Table 1: Pixel Resolutions of Different Devices.

3.2 Monitor Resolution

Table 1 shows the standard pixel resolutions for currently available PC monitors. Resolution has been increasing much more slowly than processing power and disk sizes: over the last decade, graphics workstation monitor resolution has increased only by a factor of four, from 800×600 to 1600×1200. (Over the same period, CPU speeds and hard disk sizes have increased by two orders of magnitude.) Today, even the most powerful monitor contains one order of magnitude fewer pixels than the human eye's resolution, suggesting that increased visual scalability can be attained by improved monitors. However, wall-sized displays such as AT&T's 4000×4000 pixel display are within one magnitude of the limits of human perception [1, 27].

3.3 Visual Metaphors

Visual metaphors affect visual scalability strongly. Put simply, some visual metaphors scale well in some circumstances, while others do not.

In this paper, we deal with six principal visual metaphors, which are illustrated in Figure 1 using Visual Insights' ADVIZOR[5] [22]:

Bar charts (top left in Figure 1) are collections of vertical bars arranged in a window. Two data attributes can be encoded in the bar height and color, and bars can be clustered or stacked to increase the number of attributes.

In a bar chart, the minimum possible thickness for each bar is a single pixel, as is the minimum separation between adjacent bars.[6] Assuming a window width of 1000 pixels, at maximum zoom a bar chart can display at most 500 bars. However, especially when there is little structure to index the bars, 50 bars is more realistic.

Matrix views (top right in Figure 1) [5] display one or more responses as a function of two numerical or categorical indices (in Figure 12, product and state). The view is a two-dimensional grid with rows corresponding to one index and columns to the other. Cell (i, j) contains a glyph depicting the values of one or more attributes (usually numerical) when the row index

[5]This is not to imply that other software packages do not possess similar or even more powerful capabilities.

[6]Without separation, the bar chart effectively becomes a histogram, and encoding of a data attribute in color may be impossible.

is i and column index is j. Data attributes are encoded as visual characteristics of the glyph, such as color, texture, shape and size.

Since each cell in a matrix view represents a single data element, visual scalability is governed by the number of visible cells. Cells that are 10×10 pixels are easily visible; therefore, a monitor with 1280×1024 pixel resolution can display approximately 13,000 entities. At most two orders of magnitude improvement seems possible, even in principle. Were it possible for each cell to be represented a single pixel and if space between cells were unnecessary, approximately 1,000,000 items could be displayed on a high resolution monitor.

Aspect ratios also limit matrix view scalability when the data to visualized do not map naturally to the aspect ratio of the monitor. For example, 50×40 grid better matches the screen aspect ratio than a 1000×2 grid.

Landscapes (middle left in Figure 1)[7] are three-dimensional version of matrix views. They show two-dimensional tabular data using as glyphs "skyscraper"–like towers arranged on a grid. Usually, as in Figure 1, a landscape is viewed from an angle (from straight overhead, it becomes a matrix view). The height, color and shape of the towers can potentially encode three data attributes. (This depends on the nature of the attributes. For example, numerical attributes map well onto height, somewhat well onto color and poorly onto shape. Categorical attributes map best onto color.)

Landscapes can show hundreds to thousands of data elements. Limiting factors are the number of pixels used to render each 3-dimensional glyph (typically several hundred), occlusion caused by tall bars in front obscuring short bars in the back and, as for matrix views, how well the numbers of index values match the screen aspect ratio.

Network views (middle right in Figure 1) show both characteristics of individual data elements and pairwise relationships among them [37]. Nodes correspond to data elements, whose attributes become visual characteristics such as size, color and shape. The relationships between nodes are encoded as visual characteristics of links (width, color, pattern). For example, the network view in Figure 1 shows characteristics of automobile traffic in Chicago, by zones (such as the central business district, the large node at the center). Node sizes are number of destination trips and link widths show zone-to-zone flows.

Network views can usefully display a graph with tens to thousands nodes, with strong dependence on the connectivity, number of links and inherent structure of the graph. Scalability decreases dramatically as connectivity increases, because the links connecting overplot, causing the display to become confusing [12]. Graph layout algorithms [30, 37] that attempt to minimize overplotting can overcome this to some extent. Their effect is visual accessibility to data rather than display of structure, since distances may not encode relationships between nodes. Visual scalability is limited if layout algorithms destroy or distort "real" relationships (for example, geography) among nodes.

Scatterplots (bottom left in Figure 1) can display 100,000 points or more, depending on the data pattern. The primary factor limiting scatterplot scalability is point overplotting: as the num-

[7] In ADVIZOR, a landscape view is called a *Multiscape*.

ber of points increases, points overplot, not only making structure in the data, such as trends or concentrations of points, harder and harder to identify, but also rendering access to details of the data impossible.

Histograms (bottom right in Figure 1) are essentially smoothed bar charts for data in which the bar index is ordinal or numerical. The smoothed value at any pixel of a data table depends on the smoother window size, kernel function used to do the smoothing and colors. Histograms can display 50,000 to 100,000 points, depending on the smoothing level and number of stacked colors. The smoothing itself, while an $O(N)$ operation, is a limiting factor: it requires a complete pass through the data table and potentially has a large overhead constant [28]. A smoothing calculation, for example, involving a pixel window size of 30, 40 stacked colors, a 500-pixel window width and 10,000 data rows would stress all but the most powerful workstations.

To investigate metaphor scalability we have informally consulted researchers, software developers and visual pre- and post-sales engineers who build and support customer applications to determine the largest data set that can be displayed effectively using each metaphor. The results are summarized in Table 2 for not only the six principal metaphors but also three others:

Data sheets (top of Figure 2) are scrollable text visualizations [11, 15] that provide direct access to individual data elements. A data sheet is simply a multi-column textual display that can be sorted on the basis of any column. It can display potentially hundreds of thousands of rows and tens of columns.

ParaBoxes (bottom left of Figure 2) combine Box plots and parallel coordinate plots [21].[8] Depending on data patterns, they can display hundreds to thousands of lines and ten to one hundred rows. The factor that limits ParaBox's visual scalability is overplotting caused by drawing too many lines, which leads to visual confusion.

Time tables (bottom right of Figure 2) show time-stamped, categorized events, and can display hundreds of thousands of tick marks organized into hundreds of categories.

In the absence of new techniques to increase visual scalability (see §4 for examples), the current limitations in Table 2 are also the prospective limitations.

3.4 Interactivity

The effectiveness of interactivity as a means of increasing visual scalability is substantial, but is limited primarily by inability to users navigate in high-dimensional spaces. For example, a data sheet with too many rows or columns may (like a spreadsheet) be very difficult to navigate.

Focus + context methods [8] are graphical techniques that provide a lens or other tool that magnifies the graphics in the focal area. At the same time, the magnified area is seen in context of

[8] The idea to combine the box plots and parallel coordinate plots and original implementation into a ParaBox is due to Graham Wills.

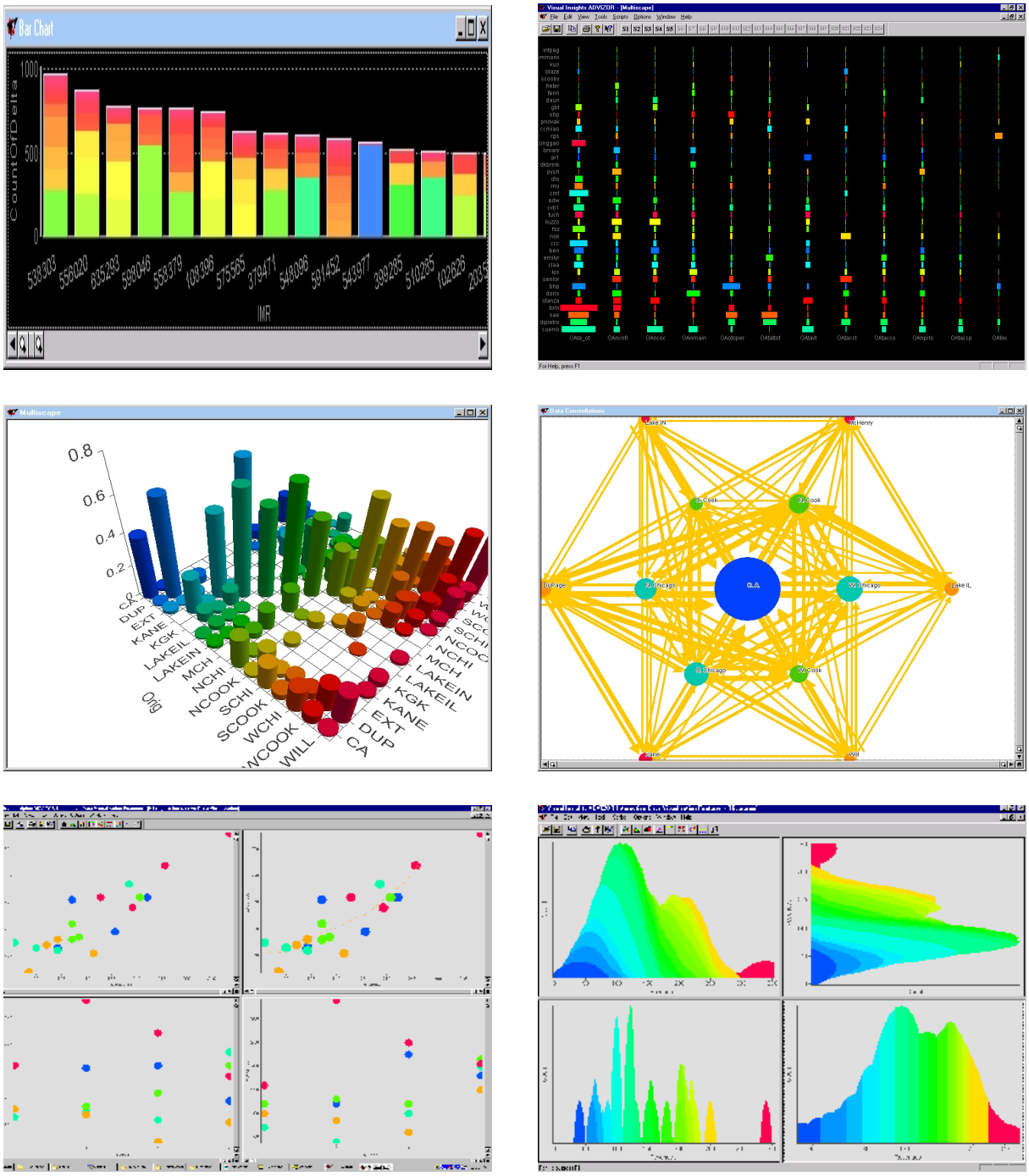

Figure 1: Principal Visual Metaphors. *Top Left:* Bar chart, and *Top Right:* Matrix view, both illustrated with software change data [16]. *Middle Left:* Multiscape, and *Middle Right:* Network view, both illustrated with zone-to-zone traffic flows in metropolitan Chicago. *Lower Left:* Scatterplot. *Lower Right:* Histogram.

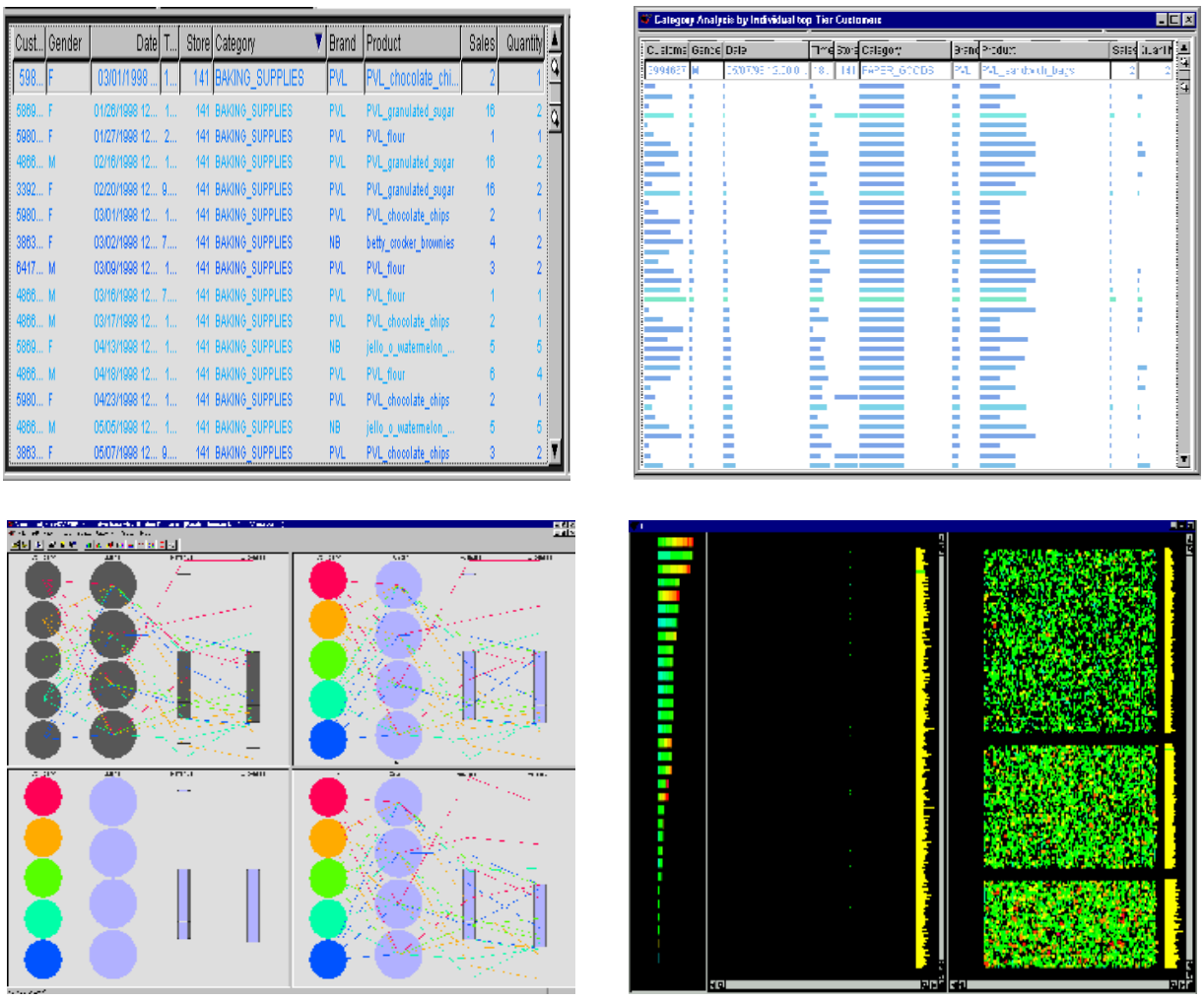

Figure 2: Other Visual Metaphors. *Top left:* Data sheet at full resolution. *Top right:* Crushed data sheet [33]. *Bottom left:* ParaBox. *Bottom right:* Time Table.

the whole, thereby helping users maintain context. In many implementations the magnification is logical rather than literal, making it possible to show more details and hence preserve access to details of the data. In typical implementations, the ratio of the lens zoom factor to normal display is approximately 5:1 [8], suggesting, in the best case, a scalability increase of perhaps one magnitude for these techniques.

Panning and zooming involve manipulating a logical viewport over a much larger graphical display. In standard implementations, zooming occurs at fixed increments, say 25%, 50%, 75%, 100%, 200%, or even 500%. Panning is realized by manipulating horizontal and vertical scrollbars. In practice, it is hard to deal with zoom factors of more than 5:1. Panning quickly becomes awkward using conventional scrollbars, which are not effective for maintaining overall context. A common technique to address this problem involves using a linked bird's eye view that shows the location of the currently visible region relative to the entire

Visual Metaphor	Database Records	Database Attributes	Limitation on Displayed Items
Bar chart	100,000 bars	10 stacked colors	Number of bars
Matrix view	n/a	2	1000 rows $\times$ 1000 columns
Multiscape	n/a	2	100 rows $\times$ 100 columns
Network view	100,000 nodes	2	Degree of connectivity
Scatterplot	100,000	2	Overplotting
Histogram	n/a	1	Number of bins
Data Sheet	100,000 rows	100 columns	Navagability
ParaBox	100 rows	10 columns	Line overplotting
Time Table	100,000 events	100 categories	Event overplotting

Table 2: Current Visual Scalability of Different Visual Metaphors.

space.

Identification and selection are interactive operations allowing the user to identify graphical entities using the mouse. Selection involves identifying a set of entities, which may then be labeled, highlighted, printed or otherwise manipulated. See Figure 8 (§4.2) for an illustration.

Automatic aggregation occurs when a user selects a set in one view that causes a linked view automatically to aggregate over the selected set. For example, linked bar charts can show categorical data effectively when selecting a bar in one chart causes an automatic aggregation for this subpopulation in the others [13].

Brushing is a standard technique whereby the user manipulates a selection tool, e.g., a rectangle, called a brush; the data items so identified are highlighted in all linked views [3].

3.5 Data Structures and Algorithms

Data structures and algorithms (discussed here) and computational infrastructure (§3.6) support and extend visual scalability, rather than create it. For this reason, our discussion of these items is brief.

For simple graphical metaphors and visual scalability problems, subtle algorithms and data structures for graphical rendering are not necessary. Modern machines simply have enough memory and are fast enough that the limits tend to be in rendering the display. This is not true, however, for complex metaphors such as network views that involve graph layout.[9] For example, a graph layout problem containing 30 nodes and 100 edges is large enough to demand very intensive computation.

[9]Or for scientific visualization problems that involve significant computation. for example, to solve partial differential equations.

By contrast, for interactive operations such as identification, selection and aggregation, data structures and algorithms frequently are the limiting factor. To be most effective interactive operations must be perceived as being (virtually) instantaneous. Human factors studies suggest three time scales [7]: (1) 10 seconds for a task; (2) 1 second for a graphical transition; and (3) 1/10 second for instantaneous changes. Using standard algorithms and data structures such as quad trees for identification and line crossing algorithms for selections, it is possible interactively to manipulate displays with 100,000 graphical entities on standard desktop machines.[10]

In many industrial data warehouses, transaction data are accumulated in large tables and then aggregated into multi-dimensional data cubes for reporting and analysis [26]. For additive measures such as sales, profits or inventory, aggregation-based techniques for navigating, slicing and dicing data cubes are highly scalable. Efficient algorithms for storage and access to large tables are available, for example, [29], as are efficient binning algorithms for computation of histograms [28].

3.6 Computing Infrastructure

The three most important aspects of computing infrastructure that affect visual scalability involve the CPU, data access, and graphical rendering:

CPU limits. As Moore's law predicts, CPU speeds continue to double every 18 months. Because, as indicated before, screen sizes and pixel resolution, are increasing much more slowly, through time strategies that trade off CPU performance for pixel resolution will increase visual scalability.

Data access and networking speeds. Network speeds are increasing more rapidly than CPU performance, doubling every nine months. Thus, it is becoming practical to access and manipulate very large data sets. Even so, the explosive growth of the Web and browser-based systems running over telephone lines has lead as well to renewed interest in visual scalability for systems with low data access rates.

Rendering rates. High performance 3D graphics cards are now common in desktop workstations. Operations involving texturing, shading, and tesselations are performed by specialized chip sets within the card. with a one order of magnitude increase in rendering performance.[11]

4 Strategies to Increase Visual Scalability

This section describes techniques to increase visual scalability, emphasizing improved visual metaphors (§4.1), interactivity (§4.2), perspectives (§4.3) and aggregation of multi-dimensional data (§4.4). For concreteness, we again illustrate using Visual Insights' ADVIZOR.

[10] On machines such as Intel's new dual processor Itanium, this may increase to 500,000 entities.

[11] The entertainment industry is driving the graphics community: interactive 3D games demand significantly higher graphics performance than is currently available.

4.1 Improved Visual Metaphors

Given that human and display limitations are within sight (see §2), improved metaphors — or combinations of them, as discussed in §4.3 — represent a significant opportunity to increase visual scalability. We now describe how refinements to the visual metaphors described in §3 (and illustrated in Figures 1 and 2) can increase scalability.

Bar charts. ADVIZOR supports color-coded stacked bars in which each bar is a sequence of up to 65 individually sized and colored vertical slices. In addition, several optimizations are employed to increase drawing speed. The rendering algorithm uses a multi-resolution drawing strategy. When sufficient screen space is available bars are drawn at full resolution with a three-dimensional boundary. As the user zooms out, the bars are drawn with progressively less detail, until they become one pixel thick. As the user zooms out further and bars become less than one pixel thick, we use an overplotting indicator and have optimized the drawing code to only draw each bar once. To avoid label overplotting, the labeling algorithm labels only every n^{th} bar, where n varies according to the number of visible bars. Labels are drawn at an angle, in order to pack more into the same screen real estate.

Zoom control (see §4.2) allows more than a 200:1 zoom, suggesting that a maximum scalability of 10,000 bars with no overplotting. In practice, zoom control and overplotting strategy can handle 100,000 bars with current machines; beyond that rendering becomes unacceptably slow.

Other strategies include flexible re-ordering of the bars (when there is no intrinsic order) [33]; in this case, the insight might be the order that emerges.

Landscapes. Multiscape uses several techniques to increase visual scalability. The first is a *water level* implemented as a translucent cutting plane that highlights towers exceeding a threshold (see Figure 3). Second, row and column sorting enable users to arrange the 3D towers to avoid occlusion problems. (However, sorting may destroy inherent structure, such as time, in rows or columns.) Third, varying symbols, such as blocks, towers, cones and levels of detail, can encode different data attributes. Fourth, predefined fixed view points, selectable using the navigation bar shown in Figure 4, and smooth animations between them make navigation easier and context changes more readily perceived. Finally, labeling algorithms drop out row and column labels that would otherwise overplot.

The effect of these techniques is a ten-fold increase in the number of data elements that can be displayed, that is, an increase approximately $\sqrt{10}$ in the number of row and column index values.

Network views. Data Constellations [37] incorporates several features to increase scalability. In order to achieve drawing speed, it uses a sequence of graph layout algorithms that progressively refine the layout. If the graph is sparse or has a regular structure that the positioning algorithms can exploit, scalability may be significantly higher. Data Constellations also uses simple graphical symbols that are easily rendered. During panning and zooming, the least significant nodes and links are dropped in order to ensure interactive drawing rates.

In networks with moderate connectivity, the effect of these features is an increase of one to two orders of magnitude in the number of nodes that can be displayed. For highly connected networks, however, there is little improvement.

Scatterplots. Three techniques are used to improve visual scalability. First, during pan and zoom operations the rendering engine selectively drops out points to ensure interactive response

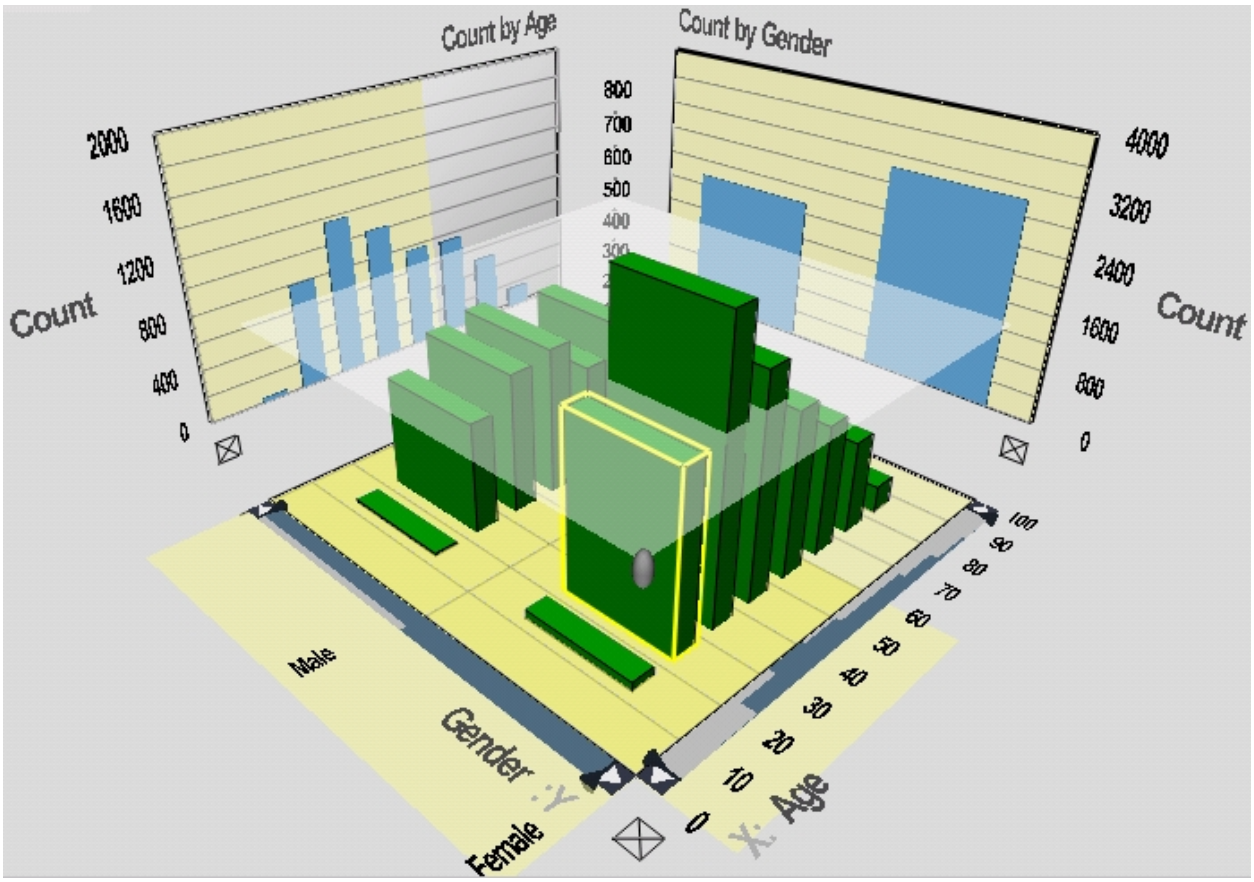

Figure 3: Multiscape view with walls, 3-dimensional scrollbars and water-level translucent cutting plane that facilitates focus on extreme data values. The data are (age,gender) counts from a sample data table. The walls contain bar charts showing the marginal totals by age and gender.

Figure 4: Multiscape's navigation bar (used in ADVIZOR/2000), which supports fixed view points and rich symbol selection.

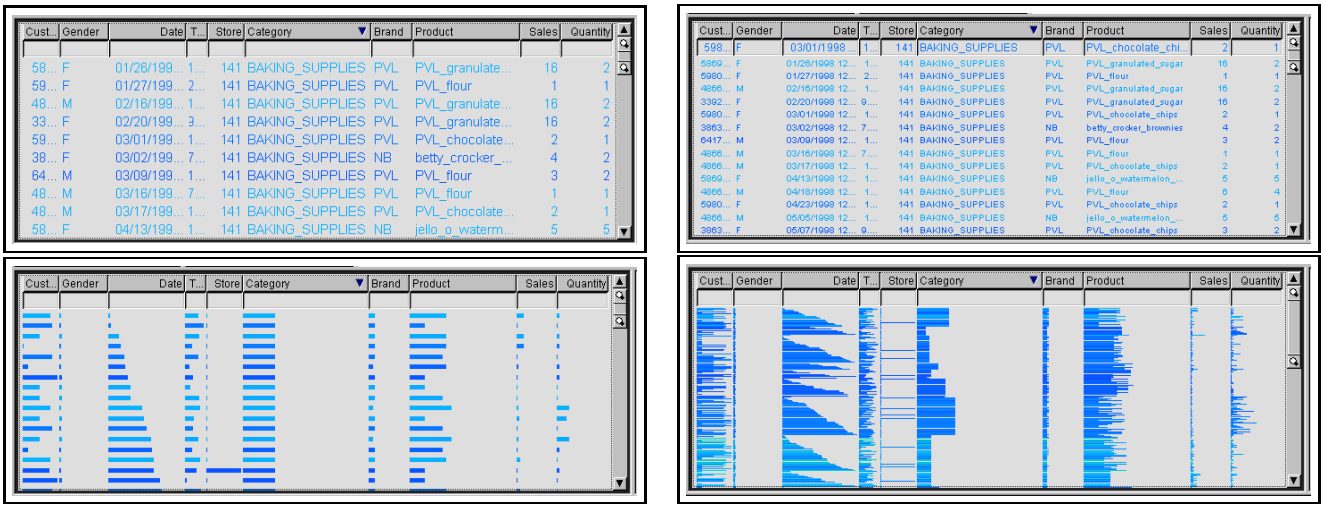

Figure 5: Data sheet is a scalable text view that smoothly transitions between text and graphics to increase view scalability. *Upper left:* full scale. *Lower right:* zoomed out, with each thin bar representing a text field.

rates. This helps the user perceive smooth and continuous motion. Second, if there is a third data attribute (beyond those corresponding the axes of the scatterplot) encoded as point color, the most significant points (as determined by the coloring variable) are drawn last, so they appear on top. Third, the rendering optimization engine drops out overplotted points that would not show anyway.

Other approaches to increase visual scalability of scatterplots include jittering [10] and of density estimation [9, 20]. Jittering increases scalability for scatterplots that repeat entries by making individual points visible. Binning using either hexagons or squares as a way to increase scatterplot scalability is proposed in [9].

Histograms. The limiting factor in histogram scalability is the smoothing calculation [28], particularly for histograms with many stacked colors. Our implementation uses drawing optimizations and specialized data structures to calculate smoothing for just those pixels that will appear on the screen.

Data sheets. To achieve visual scalability, data sheets incorporate: (1) Varying font sizes that change according to the number of rows visible, eventually switching to thin bars; (2) Bars that change their appearance according to the number of bars visible; (3) A bar-based overplotting strategy with drawing optimizations, so that each bar is drawn only once; and (4) Light-weight data structures with a virtual window into the data table for fast rendering. See Figure 5. Table Lens [31] has similar capabilities.

ParaBoxes. Scalability features include: (1) Drawing optimizations on outliers; and (2) A dialogue box for reordering the columns.

Time tables. Scalability is increased by means of: (1) Simple graphical symbols that overplot gracefully and are easy to render; (2) Zoombars for efficient panning, scrolling, zooming and navigation (see §4.2); and (3) Labeling algorithms that avoid overplotting.

Figure 6: Zoombar that supports 200–to-1 to 1000–to–1 zoom.

4.2 Exploiting Interactivity

Interactivity increases scalability in two related ways. First, focus + context devices (see §3.4), it yield a *de facto* enlargement of views. For example, a user may pan to search for features of interest in a network view (provided that the metaphor makes those features visible in zoomed out views) and then zoom in to view details of those features. Second, even without explicit use of animation [4], interactivity makes time an additional dimension of views. This exploits motion- and edge-detection capabilities of the human visual system.

We now discuss several specific devices that create interactivity.

Zooming and panning. *Zoombars* are double-edged scrollbars that provide a zoom of approximately 1000:1. The zoombar shown in Figure 6 uses a one-stage linear zoom with additional capabilities. At maximum scale on a large monitor, the two thumbs might be 1000 pixels apart, while at minimum scale they are five pixels apart. Using linear zooming, this provides a 200:1 magnification.

A discontinuity in the last few pixels increases the zoom factor by another factor of 5. At maximum zoom, by default, the implementation automatically zooms to full size. The keyboard arrow keys then move the zoomed display one unit, which may correspond to a fraction of a pixel, so that the display moves smoothly.

Other devices to facilitate zooming and panning include: (1) Continuous zoom activated by a keyboard combination, where the zoom or pan factor is controlled by mouse displacement and the display smoothly updates; and (2) Use of the “page up” and “page down” keys to double or halve the scale.

Multi-resolution metaphors. Depending on the scale, visual metaphors can be made more scalable by rendering progressively less detail as the scale increases. In bar charts as shown in Figure 7, for example, the bars are displayed with progressively less detail until they become one pixel thick. Zooming further causes the bars to overplot gracefully. Network views, scatterplots and histograms are implemented similarly. Currently, effective multi-resolution versions of matrix views and landscapes do not exist.

Selection. By interactively filtering and focusing in order to select subsets of the data (for example, a few nodes or links in a network view) and by hiding unselected items, users can reduce visual complexity and thereby increase scalability. Selection is illustrated in Figure 8, where it focuses attention on three bars within a set of linked bar charts.

Labeling algorithms. Label overplotting limits visual scalability. To reduce this problem, the views use varying font sizes and labeling algorithms that avoid overplotting. For example, in Figure 8, only every other bar is labeled.

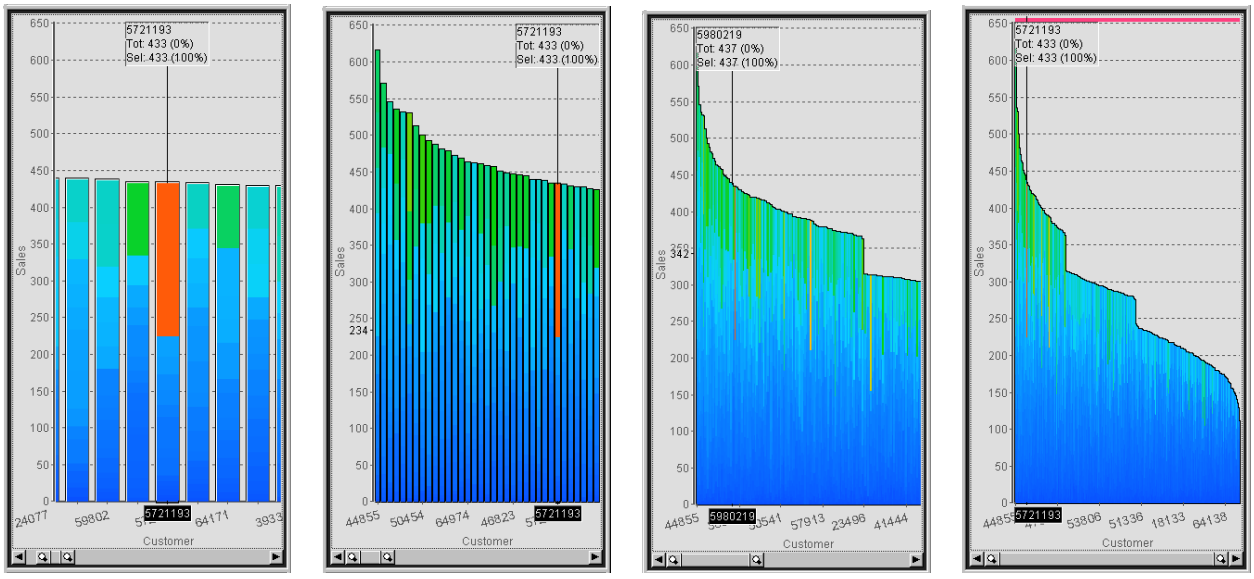

Figure 7: Bar chart scalability is increased by using levels of rendering detail and a red overplotting indicator (at the top of the view. Scalability in this case facilitates locating and then focusing attention on particular bars.

4.3 Perspectives and Visual Design Patterns

Each kind of view has strengths and weaknesses. For example, matrix, Multiscape and network views support discovery of high-level structure in data, but do not scale well in isolation. Data sheets, by their very nature provide immediate access to details of the data. Bar charts and scatterplots are primarily tools to explore and summarize the data, and are effective selectors.

To exploit the complementary capabilities of different views, and increase the scalability of all, one can construct *perspectives*, in which multiple views of the same data are displayed simultaneously, each using a different visualization strategy. More important, the views in a perspective are *linked* [6]: selection operations (see §4.2) in one view are transmitted to the others. This allows the views that summarize or access details of the data to act as filters for the views that enable understanding of high-level structure. Other systems implementing this technique include DataDesk [34], Spotfire [2], and SAS JMP [23].

Perspectives increase visual scalability by simultaneously showing a data set in different ways. Figure 9 illustrates using seven-dimensional automobile emissions data. There selection first in one scatterplot and then in a second is propagated to all four scatterplots, allowing the user to focus on a subset of the data defined by restrictions in two dimensions. See also the discussion of selection sequences in [19].

Figure 10, another perspective, displays software change data [16]. There, two bar charts showing modification requests (MRs) indexed by initiating developer and assigned developer (for repair), two pie charts showing MRs indexed by severity and status, and a scatterplot of MRs over time, collectively reveal more about the data than any one could by itself.

Perhaps the most common *visual design pattern* for perspectives [13] consists of:

A central view such as a Multiscape, scatterplot, Data Constellations or other metaphor providing

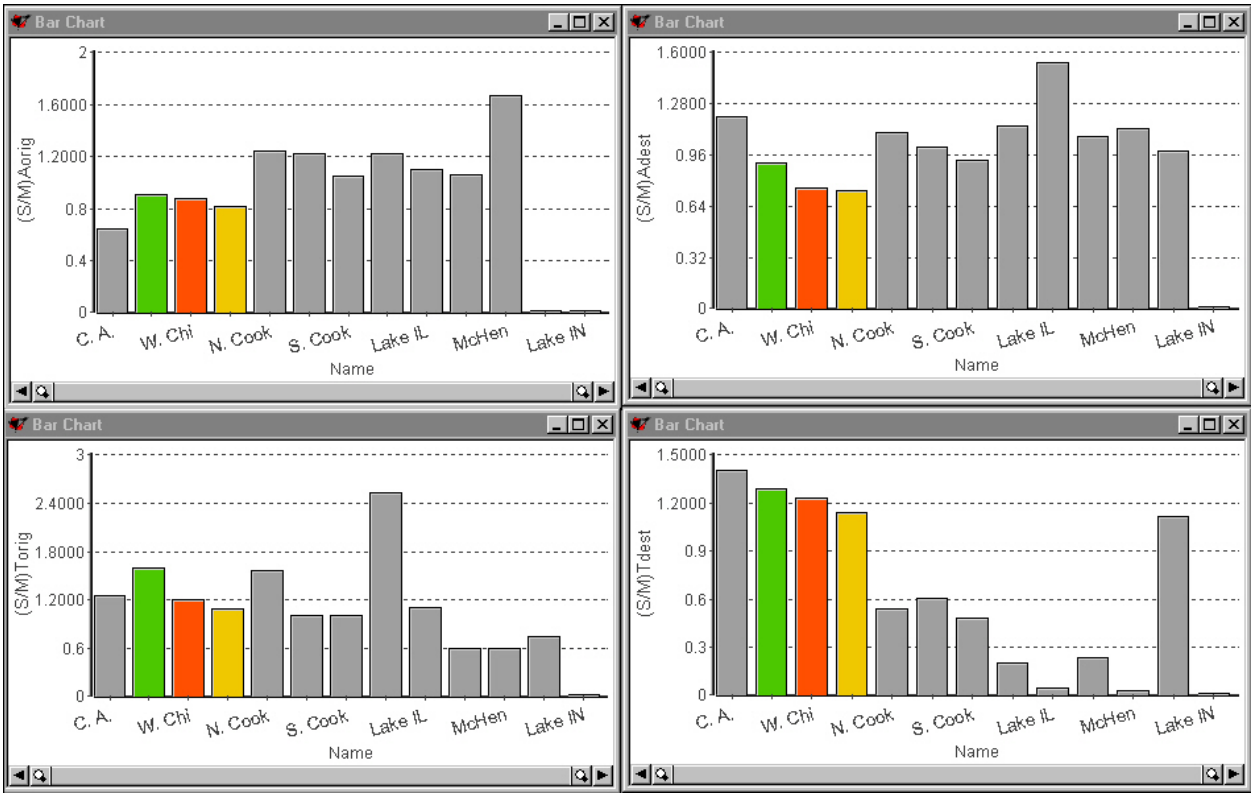

Figure 8: Selection in linked bar charts visualizing automobile (top) and transit (bottom) usage by zone of origin (left) and destination (right) in the Chicago area. Values are ratios of survey data to predictions generated by a particular model. Selection, made by the user in the upper right view and propagated automatically to the others, focuses attention on three zones in which model predictions deviate from survey values.

an overview of "response–like" attributes in the data, but in which selection of subsets of the data, especially those associated with particular values of potentially "causal" factors, may be difficult.

Supporting views such as bar charts, data sheets, histograms, pie charts, or even text listings, of "factor-like" attributes in the data that may affect the responses. Principally, the supporting views are filters enabling effective selection of subsets of the data.

The scalability problem that this design pattern addresses is visual clutter caused by too much information on the central view, which the user reduces by filtering in the supporting views.

Figure 11 fits this model well, as does Figure 12, discussed §4.4.[12] In Figure 11, which is based on the same emissions data as Figure 9, the central view depicts three pollutant responses, using a scatterplot of CO emissions against NO_x emissions, colored by the level of hydrocarbons. The two supporting views are histograms of two factors that may influence emissions, namely vehicle speed

[12]Neither Figure 9 nor Figure 10 fits the model well.

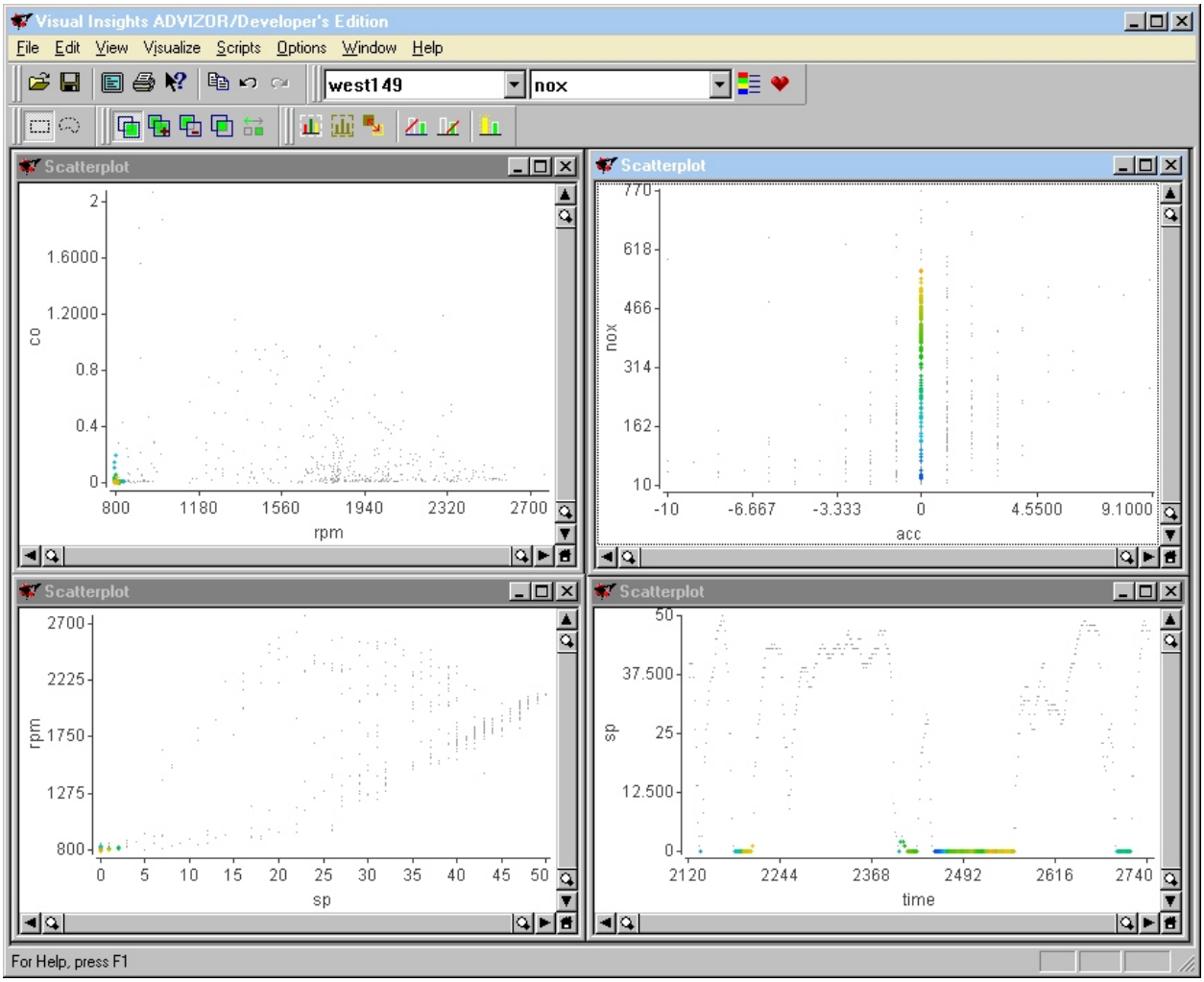

Figure 9: Linked scatterplots of 7–dimensional data on automobile behavior and associated emissions during a single trip of approximately 10 minutes. Different points correspond to measurements at 10-second intervals. The plots show CO emissions vs. engine RPM (upper left), NO_x emissions vs. acceleration (upper right), engine RPM vs. vehicle speed (lower left) and speed as a function of time (lower right). Points are colored according to the level of NO_x emissions. Selection for low speeds (lower left view) and then for zero acceleration (upper right view) demonstrates that even with these restrictions, NO_x emissions vary dramatically, underscoring the difficulty of predicting emissions.

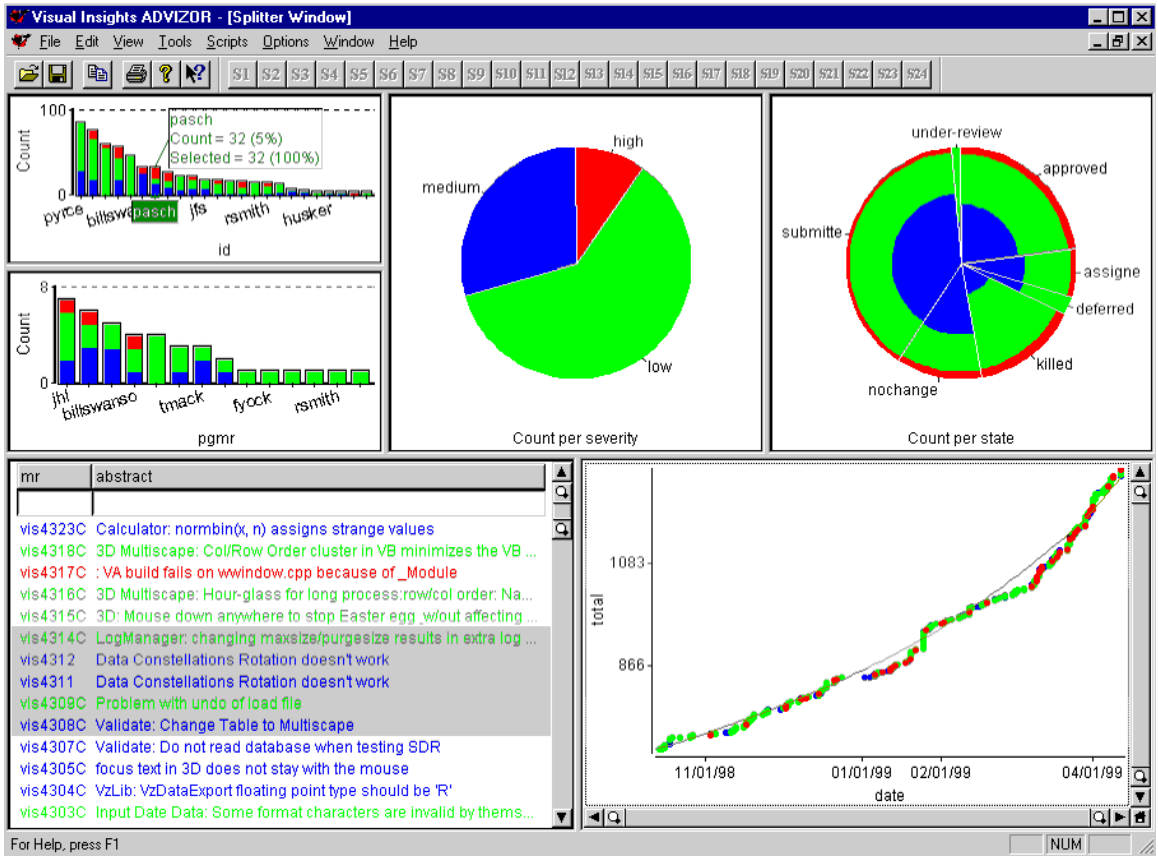

Figure 10: Perspective showing software change data described in [15]. The multiple views increase scalability by displaying five attributes of each change: date, developer, severity, status, description. In addition, a cumulative plot of the number of changes over time is shown.

and acceleration, and allow selection of data points of interest, in this case, those corresponding to high vehicle speeds (but, as shown in the scatterplot, not necessarily high emissions).

4.4 Multi-dimensional Databases: Scalability through Aggregation

Multi-dimensional databases such as Microsoft's SQL Server contain multi-dimensional structures of indexed cells created by aggregating counts (or other quantities) from relational tables. These structures are commonly called data cubes. An example for sales data is a cube containing month–by–store–by–department sales; in this case the cell `sales[Nov][Store7][Plumbing]` might contain the sum of all sales (quantity or dollar value) during November for Store7 in the plumbing department. The categorical indexing variables are called dimensions and aggregated data stored in the cells are called measures.

Often the indices are hierarchical. For example, months can be aggregated into quarters or years, stores into cities, states or regions, and departments into divisions. This hierarchical structure provides another path to visual scalability, illustrated in Figure 12 using ADVIZOR/2000.

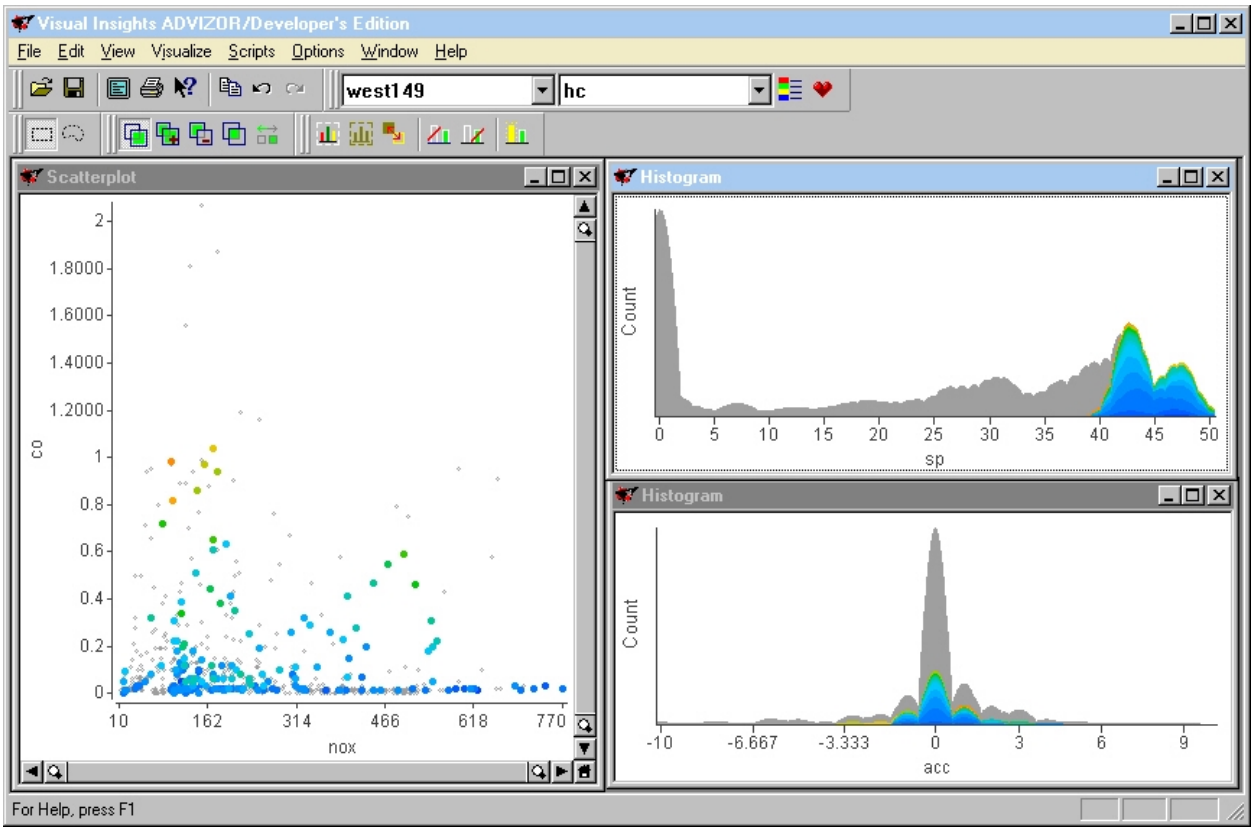

Figure 11: Visual design pattern with central and supporting views, using the same emissions data as in Figure 9. The supporting views enable easy filtering of variables that may effect the emission level responses in the central view.

This tool visualizes multi-dimensional databases using a visual design pattern (§4.3) consisting of one central Multiscape view and multiple supporting bar charts. In Figure 12, the Multiscape shows product sales by state and product, and the linked bar charts allow selection by means of either of these or an aggregation of products into product types [14].

5 Discussion

5.1 Relation to Previous Research

As noted in §1, previous analyses of visual scalability are very few. Available references include [36], which emphasizes human visual capabilities, and [24, 25], which explore limitations of pixel-based visualization techniques.

The strategies described in §4 build on previous research by the authors and collaborators, including [4, 12] on network views, [11, 15] on text visualization, [16] on perspectives (in the context of software changes) and [13, 14] on multi-dimensional data.

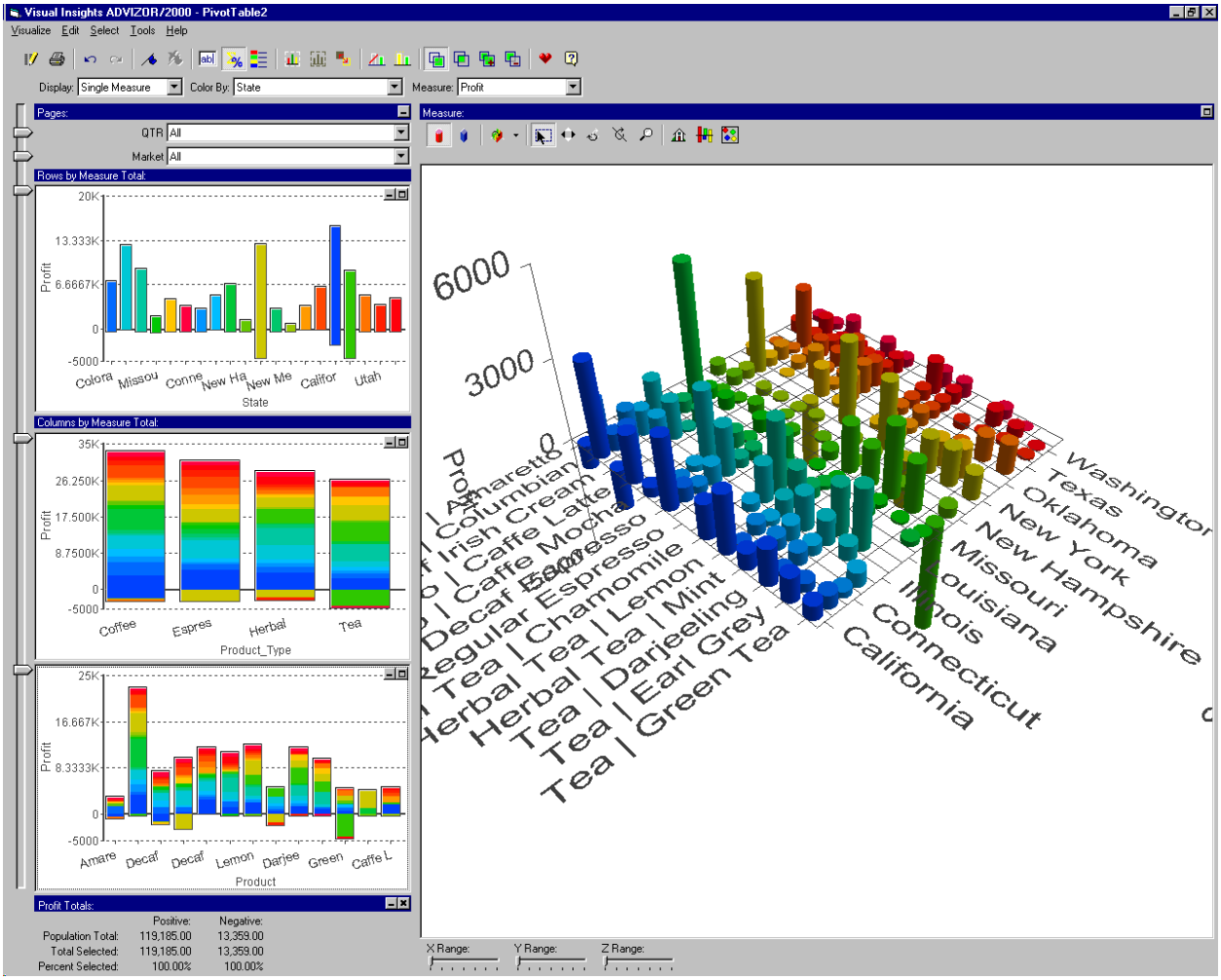

Figure 12: Data table scalability by means of aggregation, using ADVIZOR/2000. The central Multiscape view shows product sales by type and state, and the bar charts allow filtering by product (bottom), by state (top) or by product type (middle), a hierarchical aggregation of products.

5.2 Implementation

The views and perspectives presented in this paper are implemented via ADVIZOR [22] and ADVIZOR/2000 [14]. The capabilities exploited here are to create multiple individual views (matrix and landscape views, bar and pie charts, data sheets and network views), and to link these into perspectives. ADVIZOR/2000 also provides dimensional navigation, e.g., drill-up, down, and across, by tightly integrating with the database and leveraging the multi-dimensional database.

The examples using software change, transportation and emissions data described in §3 and 4 are taken from research projects conducted at the National Institute of Statistical Sciences. The concepts and questions are generic, however, and transfer readily to other settings.

6 Conclusions

In this paper, we have provided a structure for thinking about and addressing the question of visual scalability, particularly with respect to the role of visualizations as a means to access details of data. Factors that increase or limit visual scalability, ranging from human visual capabilities to hardware to visual metaphors to interactivity, were assessed in both current and prospective terms. For many factors, limits are within view, so that improved visual metaphors, interactivity and perspectives (multiple, linked views) offer the most potential to increase visual scalability. A number of strategies involving improved visual metaphors, interactivity and perspectives were presented, and their effects on visual scalability described.

Acknowledgments

A preliminary version of this paper was presented at a Workshop on Statistics and Information Technology held at the National Institute of Statistical Sciences in November, 1999.[13] We thank participants at the workshop for their comments and questions.

Many engineers and researchers have contributed to the development of ADVIZOR over the last several years. Key contributors include Ken Cox, Dianne Hackborn, John Luers, John Pyrce, and especially Graham Wills.

References

[1] J. Abello, E. Gansner, E. Koutsofios, and S. North. Large-scale network visualization. *ACM Computer Graphics*, August 1999.

[2] C. Ahlberg and B. Shneiderman. Visual information seeking: Tight coupling of dynamic query filters with starfield displays. In *SIGCHI '94 Conference Proceedings*, pages 313–317, Boston, Massachusetts, April 1994.

[13] And is available at `www.niss.org/itworkshop/presentationindex.html`.

[3] R. A. Becker, W. S. Cleveland, and A. R. Wilks. Dynamic graphics for data analysis. *Statistical Science*, 2:355–395, 1987.

[4] R. A. Becker, S. G. Eick, and A. R. Wilks. Visualizing network data. *IEEE Transactions on Visualization and Computer Graphics*, 1(1):16–28, March 1995.

[5] J. Bertin. *Graphics and Graphic Information Processing*. Walter de Gruyter & Co., Berlin, 1981.

[6] A. Buja, J. A. McDonald, J. Michalak, and W. Stuetzle. Interactive data visualization using focusing and linking. In *IEEE Visualization '91 Conference Proceedings*, San Diego.

[7] S. K. Card, S. G. Eick, and N. Gershon. *Information Visualiation Tutorial*. ACM Computer Press, Pittsburgh, PA, 1999.

[8] S. K. Card, J. D. Mackinlay, and B. Shneiderman. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufman, San Francisco, California, 1999.

[9] D. B. Carr, R. J. Littlefield, W. L. Nicholson, and J. S. Littlefield. Scatterplot matrix techniques for large n. *Journal of the American Statistical Association*, 82:424–436, 1987.

[10] J. M. Chambers, W. S. Cleveland, B. Kleiner, and P. A. Tukey. *Graphical Methods for Data Analysis*. Wadsworth International Group, Belmont, California, 1983.

[11] S. G. Eick. Graphically displaying text. *Journal of Computational and Graphical Statistics*, 3(2):127–142, June 1994.

[12] S. G. Eick. Aspects of network visualization. *IEEE Computer Graphics and Applications*, 16(2):69–72, March 1996.

[13] S. G. Eick. Visual discovery and analysis. *IEEE Transactions on Computer Graphics and Visualization*, 2000. To appear.

[14] S. G. Eick. Visualizing multi-dimensional data. *IEEE Computer Graphics and Applications*, 2000. To appear.

[15] S. G. Eick, T. L. Graves, A. F. Karr, and A. Mockus. Web-based text visualization. In W. Bandilla and F. Faulbaum, editors, *SoftStat '97 Advances in Statistical Software 6*, pages 3–10. Lucius & Lucius, March 1997.

[16] S. G. Eick, T. L. Graves, A. F. Karr, A. Mockus, and P. Schuster. Visualizing software change. *Submitted for publication in IEEE Transactions on Software Engineering*, 2000. Available at `http://home.visualinsights.com/Research/WhitePapers/vizchanges.pdf`.

[17] S. G. Eick, J. L. Steffen, and E. E. Sumner Jr. Seesoft™ —a tool for visualizing line oriented software statistics. *IEEE Transactions on Software Engineering*, 18(11):957–968, November 1992.

[18] H. Hofmann. Graphical stability of data analyzing software. In R. Klar and O. Optiz, editors, *Classification and Knowledge Organisation*, pages 36–43, Freiburg, 1997. Springer–Verlag.

[19] H. Hofmann and M. Theus. Selection sequences in manet. *Computational Statistics*, 13(1):77–87, 1998.

[20] R. J. Hyndman. Computing and graphing highest density regions. *American Statistician*, 50:120–126, 1996.

[21] A. Inselberg. Don't panic ... do it in parallel. *Journal of Computational and Graphical Statistics*, 14:53–77, January 1999.

[22] Visual Insights. Visual Insights ADVIZOR. 1999. Information available at `www.visualinsights.com/advizor`.

[23] SAS Institute. Sas jmp release 4.0. 2000. Available at `http://www.jmpdiscovery.com/`.

[24] D. A. Keim. Designing pixel-oriented visualization techniques: Theory and application. *Transactions on Visualization and Computer Graphics*, 2000. To appear.

[25] D. A. Keim and H.-P. Kriegel. VisDB: Database exploration using multidimensional visualization. *IEEE Computer Graphics and Applications*, 14(5):40–49, September 1994.

[26] R. Kimball, L. Reeves, M. Ross, and W. Thornthwaite. *The Data Warehouse Lifecycle Toolkit*. Wiley Computer Publishing, 1998.

[27] E. Koutsofios, D. A. Keim, and S. North. Visualization of large-scale telecommunications data. *IEEE Computer Graphics and Applications*, pages 33–35, May 1999.

[28] J. S. Marron and J-Q. Fan. Fast implementations of nonparametric curve estimators. *Journal of Computational and Graphical Statistics*, 3:35–56, 1994.

[29] A. W. Moore and M. S. Lee. Cached sufficient statistics for efficient machine learning with large datasets. *Journal of Artificial Intelligence Research*, 8:67–91, 1998.

[30] T. Munzner. H3: Laying out large directed graphics in 3d hyperbolic space. In *IEEE Information Visualization Conference Proceedings*, pages 2–10.

[31] R. Rao and S. K. Card. Table lens: Merging graphical and symbolic representations in an interactive focus plus context visualization for tabular information. In *Proceedings of ACM Conference on Human Factors in Computing Systems (CHI'94)*, pages 318–322, Boston, MA, April 1994.

[32] D. F. Swayne, D. Cook, and A. Buja. XGOBI: Interactive dynamic data visualization in the X window system. *Journal of Computational and Graphical Statistics*, 7(1), June 1998.

[33] A. Unwin and G. Wills. Exploring time series graphically. *Statistical Computing and Graphics Newsletter*, 9(2):13–15, 1999.

[34] P. F. Velleman. *The DataDesk Handbook*. Odesta Corporation, 1988.

[35] M. O. Ward. Xmdvtool: Integrating multiple methods for visualizing multivariate data. In *Visualization '94 Conference Proceedings*, pages 326–333, Washington, DC, October 1994.

[36] E. J. Wegman. Huge data sets and the frontiers of computational feasibility. 1999.

[37] G. J. Wills. Nicheworks – interactive visualization of very large graphs. In *Graph Drawing '97 Conference Proceedings*, New York. Springer–Verlag.