nViZn: An Algebra-Based Visualization System

Leland Wilkinson
SPSS, Inc.
233 South Wacker Drive
Chicago, IL 60606 USA
+1 312 651 3270
leland@spss.com

Dan Rope
Illumitek, Inc.
Reston, VA 20191 USA
+1 571-203-1310
dan_rope@mnsinc.com

Matt Rubin
SPSS, Inc.
233 South Wacker Drive
Chicago, IL 60606 USA
+1 312 651 3270
mrubin@spss.com

Andrew Norton
SPSS, Inc.
233 South Wacker Drive
Chicago, IL 60606 USA
+1 312 651 3270
anorton@spss.com

ABSTRACT

We describe a system, called *nViZn*, that implements a language for quantitative graphics. The structure of this system differs from existing statistical graphics, visualization, and mapping systems. Instead of treating a graphics display as a viewer for underlying data, *nViZn* treats data as an accessory to viewing a graph. *nViZn* is based on the mathematical definition of the graph of a function and uses that definition to organize data linked to the graph.

Keywords

Charts, algebra, graphics, visualization

INTRODUCTION

This paper outlines a framework called *nViZn*, which we have developed as a language for presenting graphics on the World Wide Web. Although *nViZn* is tailored to the Internet, it reflects a number of ideas that are independent of computing environment. First, it is based on an assumption that statistical procedures serve graphics; graphics are not incidental displays of statistical results, but are means for perceiving statistical relationships directly [5], [6], [11]. Second, it assumes that graphical elements are alive; wherever possible, graphical features such as points, lines, bars, legends, and axes are connected to data, metadata, or statistics in a way that allows users to drill-down, link, rotate, filter, brush and zoom directly in the display [9], [1], [16], [20], [18], [17]. Third, it is based on a formal model of graphics [21]; individual displays are not *ad hoc* visual arrangements of data, but reflect instead a quantitative or qualitative algebraic model of the variables in the display.

OVERVIEW

Figure 1 contains a diagram of the temporal model underlying *nViZn*, one of several architectural views the system. The shaded rectangles represent functional objects. The rounded rectangles represent the sets on which these functions operate. This view derives from data-flow and pipeline models devised for scientific and statistical visualization systems [3], [19].

The data for the graphic in Figure 1 are described in [4]. The climate variables (Precipitation and Growing Degree Days) were computed from an analytic method called Parameter-elevation Regressions on Independent Slopes Model (PRISM) developed in [7]. Data sets and further details can be found at www.ocs.orst.edu/prism. Carr et al. [4] used micro-gridded PRISM summaries for the time period 1961-1990 to develop graphics characterizing the spatial variation of climatic parameters within ecoregions. They associated each grid cell with an Omernik level II ecoregion [12], [13] using a point-in-polygon matching procedure. Figure 1 shows panels for three of the Omernik ecoregions (Ozark/Ouachita Appalachian Forests, Chihuahuan Desert, and West-Central Semi-Arid Praries).

The horizontal axis of each panel represents the average yearly precipitation in millimeters over the three decades. The vertical axis represents average annual growing degree days, a measure of the number of degrees in daily average temperature above 50 degrees summed over all days with a daily average temperature above 50. There are 78,766 grid cells underlying Figure 1.

A graphical system like *nViZn* needs to be able to represent these data points in real time, in a Web browser, in a distributed data environment, with instant access to associated metadata through pop-up annotations and other viewers. Any element in the graphic, including legend items, scale values, labels, smoothers, etc., can be queried for its associated metadata. *nViZn* also offers real-time controllers and widgets (sliders, buttons, list boxes, etc.) for transforming, manipulating, and selecting subsets of the underlying data.

COMPONENTS

The diagram in Figure 1 summarizes how this functionality is accomplished. The remainder of this paper is a walk through Figure 1. In each of the following sections, we will step successively through each functional object to see how it operates.

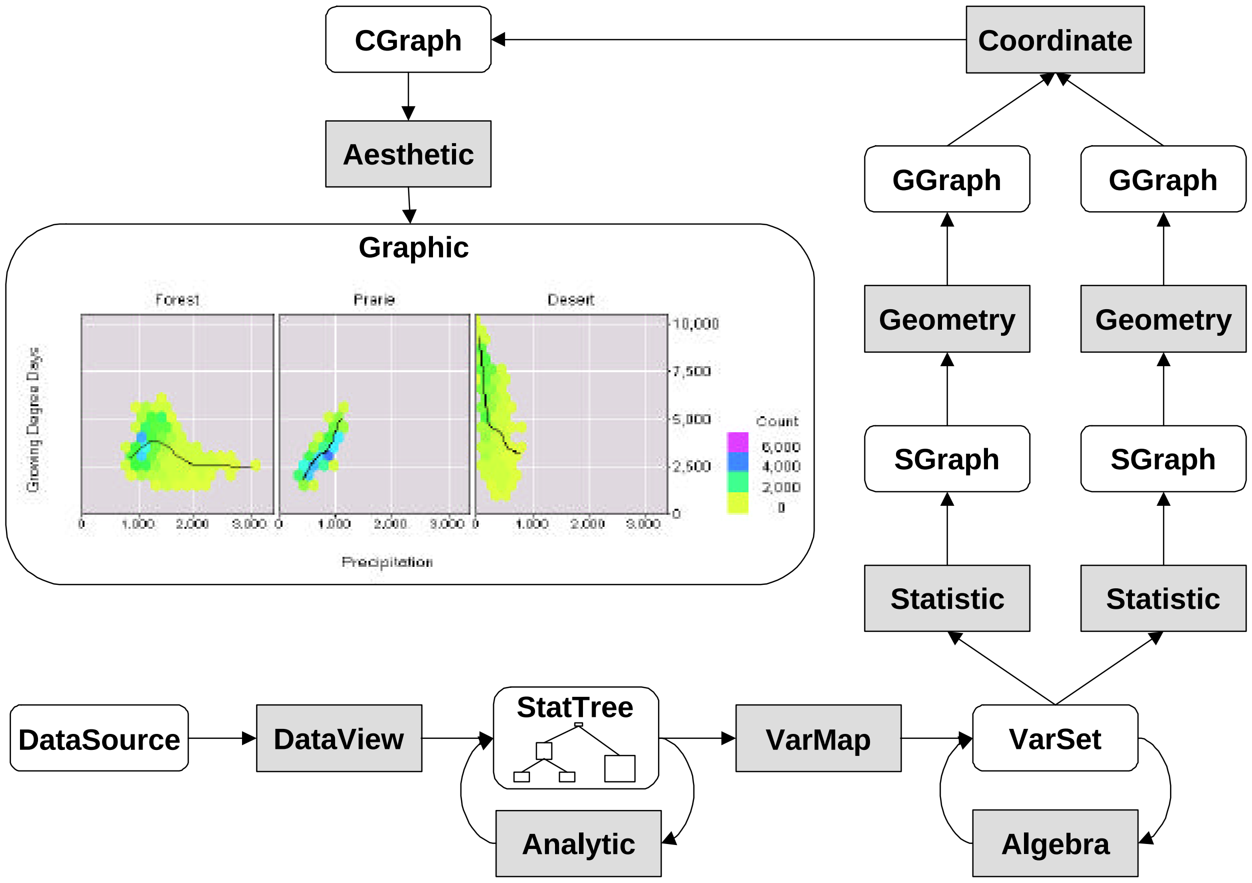

Figure 1: Data-flow view of nViZn analytical graphics system. Data for graphic are from [4]. Precipitation and Growing Degree Days computed from data in [7].

Data Views

The *nViZn* data source is an abstraction. Avoiding concrete data formats and structures encourages us to define a greater variety of graphs than is customary in relational databases or statistical packages. Having the graph organize the data, rather than having the data organize the graph, frees us from having to limit graph types to the particular structures we find in our data sources. Moreover, an abstract DataView allows us to connect our graph to heterogeneous and distributed sources of data. For example, we can collect the tuples defining 25 points in an XY plot from 25 different Web sites in a live feed to our DataView.

Not all variables found in charts are well-defined in the conventional domain of a relation. Figure 2, for example, contains a tiling of the elements of a Pearson correlation matrix. A correlation matrix is symmetric, with *k* rows/columns, one for each variable. The graphic in Figure 2, on the other hand, is two-dimensional. We require two variables (*row, column*), not *k* variables, to define this plot. DataView is an object capable of indexing datasets to fit the domain definitions of variables required in a plot.

Analytics

Analytics involve filtering, recoding, aggregating, segmenting, modeling, or summarizing data. *nVizn* Analytics are transformations that operate on an object called a StatTree. A StatTree contains a snapshot of a DataView plus, optionally, the results of dependent analyses. Because they are transformations, *nVizn* Analytics can be chained. And because their domain is a StatTree, they offer a relatively high degree of flexibility in a relatively simple object.

A tree has several advantages over other modeling data structures (such as cyclic and acyclic directed graphs). In addition to its relative simplicity, aStatTree is easily encoded in extensible markup language (XML) for use as a portable Web resource. This makes a StatTree easy to work with in a distributed network environment containing a variety of protocols.

A StatTree is a rooted tree whose nodes hierarchically alternate between data nodes containing data objects and dependent analysis nodes that identify analytic methods. This

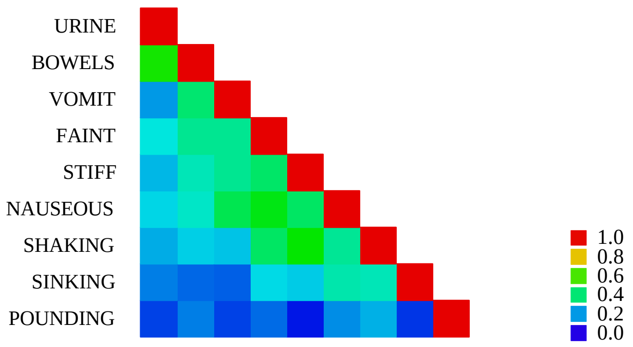

Figure 2: Tiling of correlation matrix. Color of cells represents magnitude of Pearson correlations among combat symptoms in matrix. Data are from [15].

simple structure allows us to walk a StatTree to locate a particular analysis or sequence of analyses. We can also determine both the input to an analytic method (a data object) and its output (one or more data objects).

Data nodes of the tree contain a data object identifiable by the label of the node. Instances of this object contain an array of numerical data, an array of associated string data, as well as optional resources such as case weights and metadata. Thus, a node can contain resources such as raw data, parameter estimates, fit statistics, confidence intervals, diagnostics, and model comparison statistics or information measures. All data in a StatTree must be derivable from the data at the ROOT node. Thus, descendent data nodes are either proper subsets of the ROOT data or are the results of sequences of analyses on the ROOT data.

One consequence of this architecture is that we can annotate graphics with goodness-of-fit statistics, model expressions, and other metadata from a StatTree without making additional passes through a data source to compute them. Data passes can be expensive, so it is worth collecting and persisting relatively cheap calculations even when they are not known in advance to be needed in a graphic.

Because Analytics can have StatTrees as their input and output, we may collect them in a transformation chain. Each Analytic adds one or more children to a StatTree. Thus, we can build graphics from compound analyses (e.g., cluster analyses on principal components), while maintaining case IDs, weights, and other information we need to perform brushing, linking, and sensitivity analysis.

Another benefit of transformation-chains is in handling large datasets. An abstract DataView can be used to hand Analytics chunks of data, one row or table at a time, to be aggregated by rectangular or hexagonal binning. With binning, we can process datasets with millions of cases, maintain case weights, and compute weighted statistics on the aggregates. This is how we handled the relatively bulky ecoregion data in Figure 1. The computationally intensive LOESS smoothers in Figure 1 were computed from preaggregated hex-bin data. Computing this type of smoother on a dataset this size would be otherwise impractical.

Analytics in *nVizn* currently include statistical methods like cluster analysis, regression, multidimensional scaling, correspondence analysis, and principal components analysis. *nVizn* Analytics also include organizing methods such as merging StatTree data nodes, reshaping matrices, recoding variables, partitioning variables, bootstrapping (random sampling with replacement), simple random sampling, and laying out directed and undirected graphs.

VarMap

VarMap extracts one data node from a StatTree and outputs a table called a VarSet. A VarSet is a set of variables, a matrix whose columns are variables and whose rows are instances of values on those variables. We need VarMap to make a VarSet because Algebra operates on variables, not on raw data.

VarMap finds the source table to make a VarSet through a simple StatTree addressing mechanism: a string representing the path from root to node. For example, the path ROOT/PCA/SCORES/CLUSTER/MEMBERS points to data that result from a cluster analysis on principal components. StatTree paths encapsulate what has been done to data before graphing. The StatTree path for the graphic in Figure 1 is ROOT/AGGREGATE/AGGREGATION. The AGGREGATION data object contains the coordinates and counts for the hex bins.

Algebra

The graph $G = \{(x, f(x)) : x \in \mathbf{R} \; and \; f(x) = e^{-x^2}\}$ is a subset of $\mathbf{R}^2$. To display G, we choose a bounded region of $\mathbf{R}^2$, $F = [x_{min}, x_{max}] \times [y_{min}, y_{max}]$, and we physically represent the set of points $P = F \cap G$ by choosing a coordinate system and making a graphic with ink or some other perceivable medium.

Graphics algebra provides a method for specifying F (which we call a *frame*) when we wish to construct a graphic based on some function of a set of data. Wilkinson [21] presents three algebraic operators called *cross* (*), *nest* (/), and *blend* (+), together with the rules for their use. They are derived from the set operators *product* ($\times$), *discrete union* ($\sqcup$), and *union* ($\cup$), respectively. We use *cross* to construct a graphic of the error function in the example at the beginning of this section. The algebraic expression for the frame containing the bounded graph P is $x*y$, where x is the bounded domain of the function and y is its bounded range.

Graphics algebra is symbolically evaluated. For example, the expression $a*(b+c)$ is equivalent to $a*b + a*c$; *nViZn* produces the same graphic when presented with either expression. Wilkinson [21] discusses the procedure for doing this symbolic evaluation, which resembles normalization of relational query expressions.

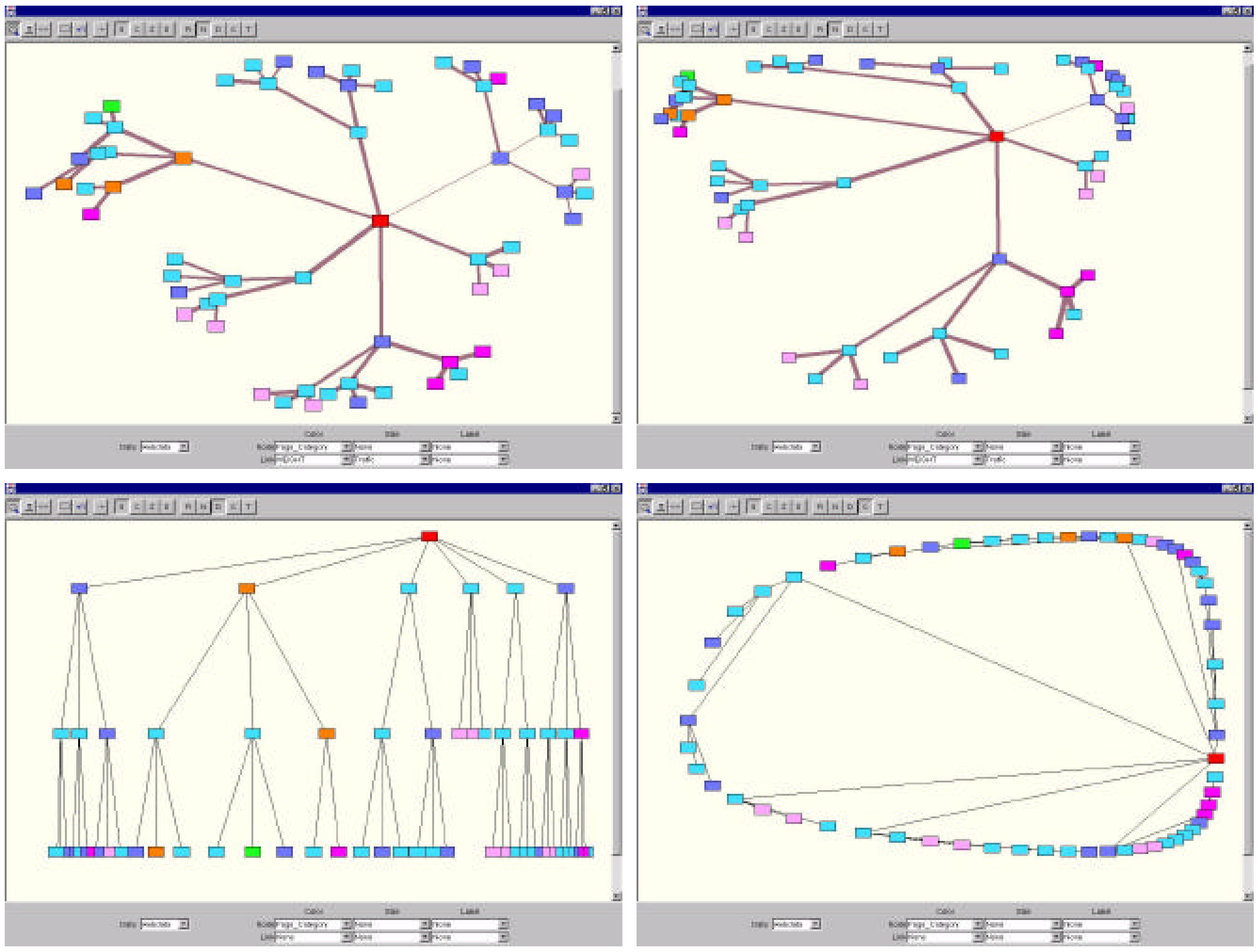

Figure 3: Lensing coordinate transformations on Web data.

Statistics

The Statistics component of *nViZn* contains functions that receive a VarSet from Algebra and output a statistical graph, called an SGraph. The most familiar statistical graphs are location statistics such as means. Statistical graphs also include confidence regions, smoothers, densities, directed graphs such as trees, and other functions. The term *statistical* is something of a misnomer, since our requirement for this component is only that Statistics output a unique tuple or set of tuples or collection of sets of tuples for each input tuple. This more general definition of an SGraph allows *nViZn* to produce a wide variety of graphics not limited to statistical charts.

Figure 1 demonstrates that a frame may include more than one SGraph. Our ecoregion example includes a hex-bin density and a LOESS smoother. Each is computed from the same VarSet. This architecture is ideally suited for a multithreaded environment in which certain tasks can be handled simultaneously. The possibility of more than one SGraph in a frame is one of the obvious ways *nViZn* differs from a charting program. Standard charts have only one or two hard-wired graphical elements per frame.

Computing statistical values requires a lot of housekeeping. We must not only handle sample weights and missing data, but we must also carry along the pointers necessary to link geometric components to data. If we compute a box plot, for example (see Figure 4), we need to maintain a list of cases in the central box, whiskers, and possible outliers so that a user can brush these objects and link them to other graphs in real-time. If we compute a tessellation, we need to maintain enough information to compute the perimeter or area of a polygon if requested. Doing this in a general and efficient way, for linear, order and other statistics, while allowing for missing data and sample weights, is nontrivial.

The hex-bin Statistics element is not the same as the hexagon binning Analytic. If we eliminated the hexagon binning Analytic from our specification, we would have to compute the hex-bin Statistic from the raw data. This would not have taken more time (since the same algorithm is involved in both computations), but it would have prevented us from computing the LOESS smoother in reasonable time. Instead, we pre-computed the bins in an Analytic and then used the bin counts and locations to compute both the hex-bin and LOESS Statistics. The same trade-offs exist for other Statistics and Analytics. Where we choose to locate the computations depends on the functionality we want in our application.

Geometry

Geometry converts an SGraph (a statistical graph) to a GGraph (a geometric graph). The sub-classes of GGraph include *point*, *line*, *area*, *bar*, *histobar*, *schema*, *tile*, *contour*, *path*, and *link*. With the *point* geometry object, we can represent a point estimate of a mean with a dot. With the bar geometry object, we can represent the same point estimate by locating one end of the bar at the coordinates of the point. The graphic in Figure 1 employs two Geometrics: *line* (for the LOESS smoother) and *tile* (for the hex-bins).

Sometimes, Geometry cannot produce a GGraph from a particular SGraph it has been given. Undefined instances (e.g., a histobar of a tree) result in null objects. These instances are surprisingly rare. As elsewhere in the *nViZn* system, modularity of function and orthogonality of design increase the potential output of the system. This design strategy also encourages us to think more broadly about graphical representation.

Coordinates

We are accustomed to seeing graphics in rectangular coordinates. Sometimes, as with pie charts, we are accustomed to polar coordinates. We rarely expect to see bar charts in spherical coordinates, however, or time series charts in polar. Mathematicians, geographers, and spatial statisticians are more inclined to transform their viewing space, but others rarely encounter geometric objects in other coordinate systems.

nViZn locates coordinate transformations in a separate object and makes them work on most geometric graphs. Coordinates convert one or more geometric graphs (GGraph) to a composite graph (CGraph). A CGraph embeds one or more geometric graphs in a single frame and its associated coordinate system. The coordinate system used in Figure 1 is rectangular. It embeds the density and smoother graphics in the frame.

Figure 3 shows a lensing coordinate transformation [10] in *nViZn.* The upper left panel of the figure shows an unrooted tree graph. The upper right panel shows the tree distorted to reveal local detail in the lower left area of the tree. The lower left panel shows the same tree displayed in rooted form, with the chosen root at the top. The lensing transformation is applied to the bottom-middle of the tree, to reveal local detail in that area. Finally, the bottom right panel shows a circular layout of the same graph with the lensing focus in the upper left region of the graph.

In all cases, the lensing tool works in real time as it is moved around the display. This functionality works because the coordinate transformation is applied only to the geometric objects in the graph. No other aspect of the specification changes as the lensing tool is moved to a new focus.

Aesthetics

Aesthetics convert a composite graph (CGraph) to a perceivable graphic. When we colloquially call a chart a graph, we are really speaking of the realization of a CGraph. A CGraph is a mathematical graph; it is not visible or perceivable. A graphic, on the other hand, is perceivable in some sense.

nViZn includes a variety of Aesthetics that extend the work of Bertin [2]. These are *position*, *size*, *shape*, *rotation*, *color*, *texture*, *blur*, and *transparency*. In addition, *nViZn* Aesthetics includes an attribute not generally thought of as an aesthetic or visual variable: a *label*. A *label* is a text object glued to an element of a graph. In a graphic, a *label* functions like a color, texture, or other attribute to make a graph perceivable to a reader.

The abstraction and localization of Aesthetics in *nViZn* yields some interesting behaviors. First, *nViZn* can construct tables of numerals or text by using a label attached to an invisible geometric element such as a point or tile. Second, a brushing event can be attached to any attribute such as a label, color, rotation, or blur. Using a brush in one frame, for example, can cause points in another linked frame to show their labels, change their color, rotate, or blur. Third, *nViZn* can be used to construct graphics that do not even remotely resemble XY plots. These include such images as appear in [8] and [14].

Controllers

A Controller is an object that connects a user gesture to a function. For example, a brush is a controller that wires a user-manipulatable brushing region (usually a rectangular brush tool) to an Aesthetic through a graph-subsetting function. In subsetting a graph, we select a region that defines a subset of the values on one or more variables. When we drive this back through the pipeline, we select a subset of our VarSet. Any Frame that is dependent on the same VarSet will receive brushing messages identifying elements in the subset and these identifiers will be mapped to a selected Aesthetic.

nViZn has over 30 controllers that allow a programmer to connect functions in the graphics system to user widgets such as sliders, list boxes, buttons, and modal cursor tools. These controllers extend the scope of *nViZn* beyond visualization, making it a system for manipulating as well as viewing data.

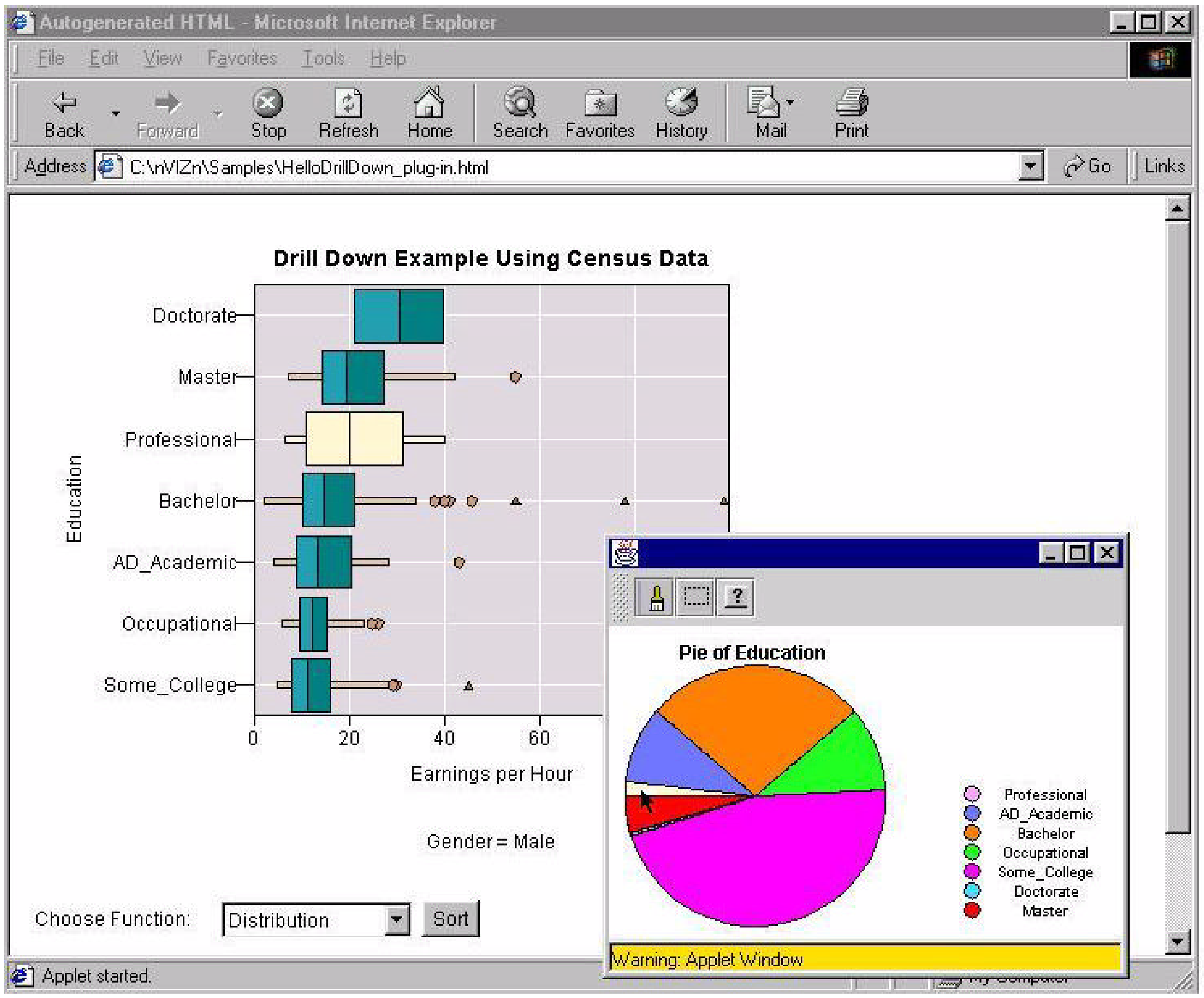

Figure 4: Drill Down and Brushing controllers. Third quartile of earnings by Males with bachelor degrees is highlighted in histogram of all earnings.

Figure 4 shows drill-down and brushing controllers applied to data from the 1990 US Census. The main graphic consists of a set of box plots of the distribution of hourly earnings among different educational groups. One controller at the bottom of the main window allows the user to select the type of graphic to be displayed. Figure 4 selects Distribution (a box plot) as opposed to a mean, median, or other point statistic.

The Sort button is attached to a controller that causes the educational categories to be sorted by hourly earnings. This facilitates interpretation of the earnings data across categories.

A drill-down controller was used to select Males only for the main graphic. Prior to this display, the user clicked on the males category in the aggregate display and the result was to filter out Females.

Finally, a brushing controller was used to link the pie chart window with the box plot window. This controller was set to use color (white) to highlight the brushed category (Professional). We see that a relatively small proportion of the population falls within the Professional postgraduate category. By selecting the Professional sector of the pie chart, we highlight the Professional box in the main display.

The brushing controller works from either window. We could, for example, point to one of the outliers at the right end of the box plot for the Bachelor group and find its corresponding location in the Pie chart. Any other chart window, whether or not it involved earnings or educational category, could be linked similarity.

CONCLUSION

nViZn defines, organizes, and constructs a graphic in graph-world rather than data-world. This might strike those accustomed to relational database or object database worlds as somewhat peculiar. As Wilkinson [21] stated, however, "These definitions are embedded in the mathematical history that determined the evolution of statistical charts and maps." In short, we begin by considering what is the range and what is the domain of a graph underlying a graphic.

From there, we recurse our definitions until we reach a specification of data underlying the graphic. For that specification, we construct an abstract DataView and link the graphic to our data.

ACKNOWLEDGMENTS

We thank Jack Noonan, Ed Hamburg, and Mark Battaglia for supporting this research.

REFERENCES

1. Becker, R.A., and Cleveland, W.S. Brushing scatterplots. *Technometrics*, 1987 (*29*), 127-142.
2. Bertin, J. *Semiologie Graphique*. Paris: Editions Gauthier-Villars, 1987. English translation by W.J. Berg as *Semiology of Graphics*, Madison, WI: University of Wisconsin Press, 1983.
3. Buja, A., Asimov, D., Hurley, C., and McDonald, J.A. Elements of a viewing pipeline for data analysis. In W.S. Cleveland, and M.E. McGill (Eds.), *Dynamic Graphics for Statistics*. Monterey, CA: Wadsworth, 1988, 277-308.
4. Carr, D.B., Olsen, A.R., Pierson, S.M, and Courbois, J.P. Boxplot variations in a spatial context: An Omernik ecoregion and weather example. *Statistical Computing & Statistical Graphics Newsletter*, 1999 (*9*), 4-13.
5. Chambers, J.M., Cleveland, W.S., Kleiner, B., and Tukey, P.A. *Graphical Methods for Data Analysis*. Monterey, CA: Wadsworth, 1983.
6. Cleveland, W.S. *The Elements of Graphing Data*. Summit, NJ: Hobart Press, 1985.
7. Daly, C., Neilson, R.P. and Phillips, D.L. A Statistical-topographic Model for Mapping Climatological Precipitation over Mountainous Terrain. *Journal of Applied Meteorology*. 1994, (*33*), 140-158.
8. Eick, S.G., Steffen, J.L., and Sumner, E.E. SeeSoft -- a tool for visualizing line oriented software statistics. *IEEE Transactions on Software Engineering*, 1992 (18), 957-968.
9. Fisherkeller, M.A., Friedman, J.H., and Tukey, J.W. PRIM-9: An interactive multidimensional data display and analysis system. SLAC-Pub-1408. Stanford, CA: Stanford Linear Accelerator, 1974. Reprinted in W.S. Cleveland, and M.E. McGill (Eds.), *Dynamic Graphics for Statistics*. Monterey, CA: Wadsworth, 91-109.
10. Furnas, G.W. Generalized fisheye views. *Human Factors in Computing Systems: CHI 86 Conference Proceedings*. New York: ACM Press, 1986, 16-23.
11. McDonald, J.A., and Pedersen, J. Geometric abstractions for constrained optimization of layouts. In A. Buja, and P.A. Tukey (Eds.), *Computing and Graphics in Statistics*. NewYork: Springer-Verlag, 1991, 95-105.
12. Omernik, J.M. Ecoregions of the coterminous United States. *Annals of the Association of American Geographers*, 1987 (*77*), 118-25.
13. Omernik, J.M. Ecoregions: a spatial framework for environmental management. In W.S. Davis, and T.P. Simon (Eds.), *Biological Assessment and Criteria: Tools for WaterResource Planning and Decision Making*. Boca Raton, FL: Lewis Publishers, 1995, 49-62.
14. Rao, R., and Card, S.K. The table lens: Merging graphical and symbolic representations in an interactive focus+context visualization for tabular information. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, 1994 (*18*), 318-322.
15. Stouffer, S.A., Guttman, L., Suchman, E.A., Lazarsfeld, P.F., Staf, S.A., and Clausen, J.A.. *Measurement and Prediction*. Princeton, NJ: Princeton University Press, 1950.
16. Stuetzle, W. Plot windows. *Journal of the American Statistical Association*, 1987 (*82*), 466-475.
17. Swayne, D.F., Cook, D., and Buja, A. XGobi: Interactive Dynamic Data Visualization in the X Window System. *Journal of Computational and Graphical Statistics*, 1998 (*7*), 113-130.
18. Tierney, L. *LispStat*. New York: John Wiley & Sons. 1991.
19. Upson, C., Faulhaber, T., Kamins, D., Schlege, D., Laidlaw, D., Vroom, J., Gurwitz, R., and vanDam, A. The application visualization system: A computational environment for scientific visualization. *IEEE Computer Graphics and Applications*, 1981, (9), 30-42.
20. Velleman, P.F. *Data Desk*. Ithaca, NY: Data Description Inc.. 1988.
21. Wilkinson, L. *The Grammar of Graphics*. New York: Springer Verlag, 1999.