Original Article

AutoVis: Automatic visualization

Graham Wills[a] and
Leland Wilkinson[b,*]

[a]SPSS Inc., 233 South Wacker, Chicago, IL 60606, USA.
[b]SYSTAT Inc., 225 W Washington St., Chicago, Illinois 60606, USA.
E-mails: gwills@spss.com;
leland.wilkinson@systat.com

*Corresponding author.

Abstract AutoVis is a data viewer that responds to content – text, relational tables, hierarchies, streams, images – and displays the information appropriately (that is, as an expert would). Its design rests on the grammar of graphics, scagnostics and a modeler based on the logic of statistical analysis. We distinguish an *automatic* visualization system (AVS) from an *automated* visualization system. The former automatically makes decisions about what is to be visualized. The latter is a programming system for automating the production of charts, graphs and visualizations. An AVS is designed to provide a first glance at data before modeling and analysis are done. AVS is designed to protect researchers from ignoring missing data, outliers, miscodes and other anomalies that can violate statistical assumptions or otherwise jeopardize the validity of models. The design of this system incorporates several unique features: (1) a spare interface – analysts simply drag a data source into an empty window, (2) a graphics generator that requires no user definitions to produce graphs, (3) a statistical analyzer that protects users from false conclusions, and (4) a pattern recognizer that responds to the aspects (density, shape, trend, and so on) that professional statisticians notice when investigating data sets.

Information Visualization (2010) **9,** 47–69. doi:10.1057/ivs.2008.27;
published online 18 December 2008

Keywords: automatic visualization; statistics; visual analytics

Introduction

As John Tukey frequently taught, statistical practice is an endless cycle of exploration, analysis, and revision.[1–3] Exploratory data analysis (EDA) is not a search for rules; it is a search for surprises. Modern visual analysis extends Tukey's idea. Its premise is that more can be learned from the visual analysis of raw data than from blind summarization.

Engaging Tukey's cycle begins with a paradox, however. How can we explore if we do not know where to look? In the traditional environment of statistical consulting, this paradox is resolved by the client. The first question statisticians ask a client is how the data were collected. In answering that question, a client tells a statistician about the ideas that generated the data as well as all the particulars of the collection itself. Sometimes, however, an analyst has no client. Data are *given* and the analyst *receives* (this condition reflects the Latin meaning of *data* – givens). In such circumstances, an analyst begins in the dark.

This paper offers one approach for providing some initial light so an analyst can enter the exploratory loop. We call this approach *automatic visualization* and the software resting on it an automatic visualization system (AVS). AutoVis is a program that provides a first view of a data source consisting of text, relational tables, hierarchies, images, or other forms. We do our best to make that view informative and consistent with the layout and methods an expert statistician would employ.

We are not the first to use the term *automatic visualization*. It is frequently employed to describe specific methods, such as graph layout. By contrast,

Received: 20 June 2008
Revised: 10 October 2008
Accepted: 21 October 2008

the related work we acknowledge here involves the more general, more difficult problem: visualizing data without presupposing a model. The pioneering work in this area was done by Jock Mackinlay in his Stanford dissertation.[4] Mackinlay drew on the work of Codd (computer science), Bertin (geography), Stevens (psychology), and the Tukey Bell Labs group (statistics) to craft a system that could display intelligently the contents of relational tables.

Steven Roth's group at Carnegie-Mellon was the other early stimulus for automatic visualization research.[5,6] As a psychologist, Roth drew heavily on cognitive theory for developing visualization strategies in his software. His system for automated graphics and explanation (SAGE) was able to graph intricate data sets, such as Minard's map of Napoleon's Russian campaign.

Others followed these researchers with systems for intelligent or 'smart' graphics.[7–12] In addition, there have been systems designed around displaying large automated visualizations, such as the CAVE,[13] Information Mural[14] and ImmersaDesk.[15] These environments are especially suited to data-rich displays and anticipate further development of automated systems to populate large displays.

Our goal in this project is somewhat different from prior work, however. We seek a beginning, not an end. Tools like AutoVis must be accompanied by tools for detailed interaction with data, including brushing, linking, sorting, and other actions introduced by Tukey's Bell Labs group and widely used today. In our view of automatic visualization, we wish to cast enough light on data to help an analyst to commence, but not enough light to help an analyst to conclude. Our task is as much concerned with what not to do (showing patterns that could lead to false conclusions about random data) as with what to do (showing significant relationships that may not be apparent under conventional analyses). We believe a statistical graphics viewer should stimulate thought, not suppress thought.

Design

Displaying a data source requires several processing stages. First, we must *recognize* the type of file or data source we are given. Second, we must *analyze* the structure of the data. Third, we must *prioritize* the views derived from our analyses in order to present the most interesting results. Finally, we must *summarize* the structure of the data source in a single display. We address each of these problems in sequence.

Recognizing

Our first tactic in recognizing a data source is to look at its signature. Proprietary file extensions are easiest. They are accompanied by standard formats that we can parse more or less straightforwardly. These formats tell us something about the structure of the file as well. We know that a word processing file is more likely to contain free text, for example. A spreadsheet file is more likely to contain one or more tables. Some formats, such as Microsoft Word files, may be so complicated and general as to tell us almost nothing about the structure within.

Text files are also difficult. They may contain free text or they may contain tables encoded in a comma-separated, space-separated, or tabbed format. Or, they may use MIME or other types of encoding to encapsulate more complex structures. Consequently, we pursue a sequential strategy. We parse them as if they were tables (looking for an optional first row of variable labels) and if that fails, we parse them as free text. Our default column separator is a space, but we also look for commas, tabs, and other delimiters.

Image files are simple, because they have industry-standard formats. Raster files are ready for analysis; we decompose them into color histograms. Vector files must be rendered into buffered image maps so that we can process them as raster.

Sometimes our data are not contained in files. They may be embedded in other structures, or presented as data streams from a URL or other source. With an HTML or XML source, we are fortunate to have tags to guide us. In other cases, we follow a sequential strategy similar to the one we use for files. With less information regarding internal structure, however, we are more likely to fail.

Analyzing

Once we know the type of data we have, we need to decide on an appropriate analytic strategy. This strategy rests on a sequence of steps outlined in Wilkinson,[16] called Grammar of Graphics (GoG). This system consists of a functional data flow computed by the methods in seven classes – Variables, Algebra, Scales, Statistics, Geometry, Coordinates, and Aesthetics. Each class contributes one method and the type of graph produced is a functional chain of the methods. These seven functional components comprise a complete order. It is claimed that they cannot be computed in any other order without producing meaningless results. Appendix B outlines this system. We now describe how this system is used in AutoVis.

1. *Variables*. First, we must determine whether a variable is continuous or categorical. (Variables and their types are defined formally in Appendix A.) For tables, we consider each column as a variable. We classify columns containing at least one non-numeric string as categorical. For columns containing only numbers, we tally numeric values and test for granularity by counting the number of discrete values. When the value tally is small, we declare a column to be categorical. Otherwise it is declared continuous.

   Other data sources are treated differently. For free text data, we consider all elements to be instances of a categorical variable. Our analysis is based on counts of instances. For images, we treat pixel values as drawn

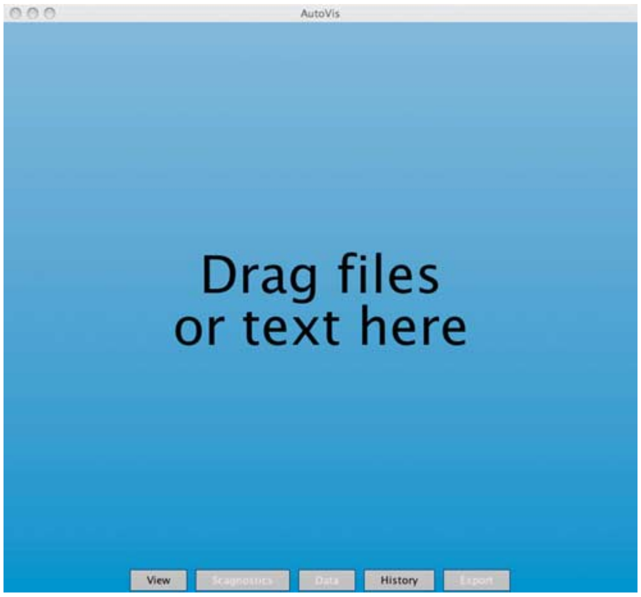


Figure 1: Opening screen (plus user manual). AutoVis has no menus or modal GUIs.

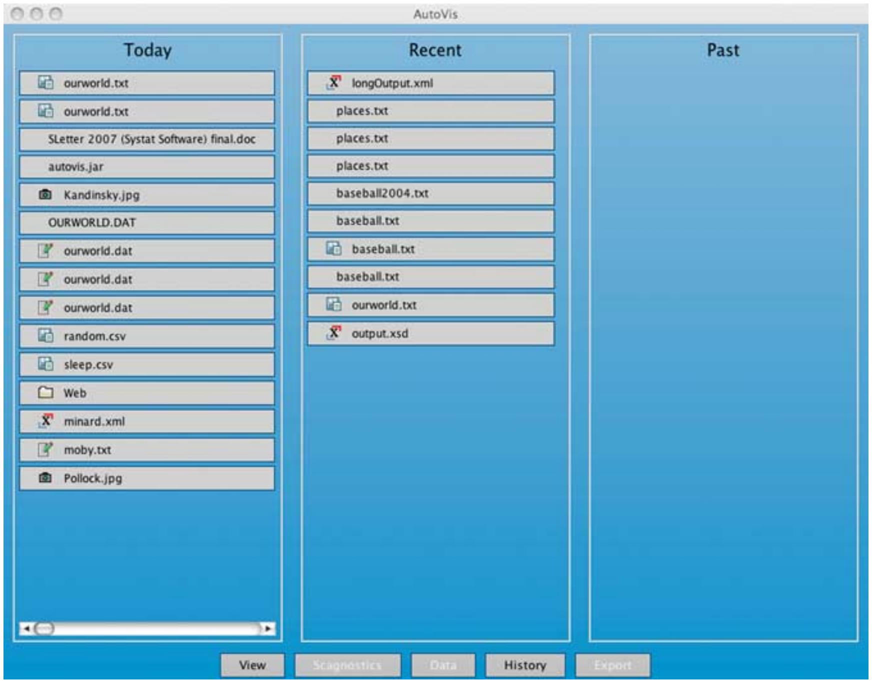


Figure 2: AutoVis history view. The three panels contain files listed by recency.

from continuous variables. We type-cast similarly for other data sources.

2. *Algebra*. The graphs in AutoVis are produced from tuples. These tuples are defined by the *cross* operator in the GoG algebra. Appendix A details this operation. We plan to incorporate the other two operators (*nest* and *blend*), but this will require more refined analysis of the structure of tabular data.
3. *Scales*. Next, we need to determine scales on which graphs will be displayed. We iteratively apply Tukey's *ladder of powers*[1,17] in order to make 1D distributions as symmetric as possible. For an estimated power

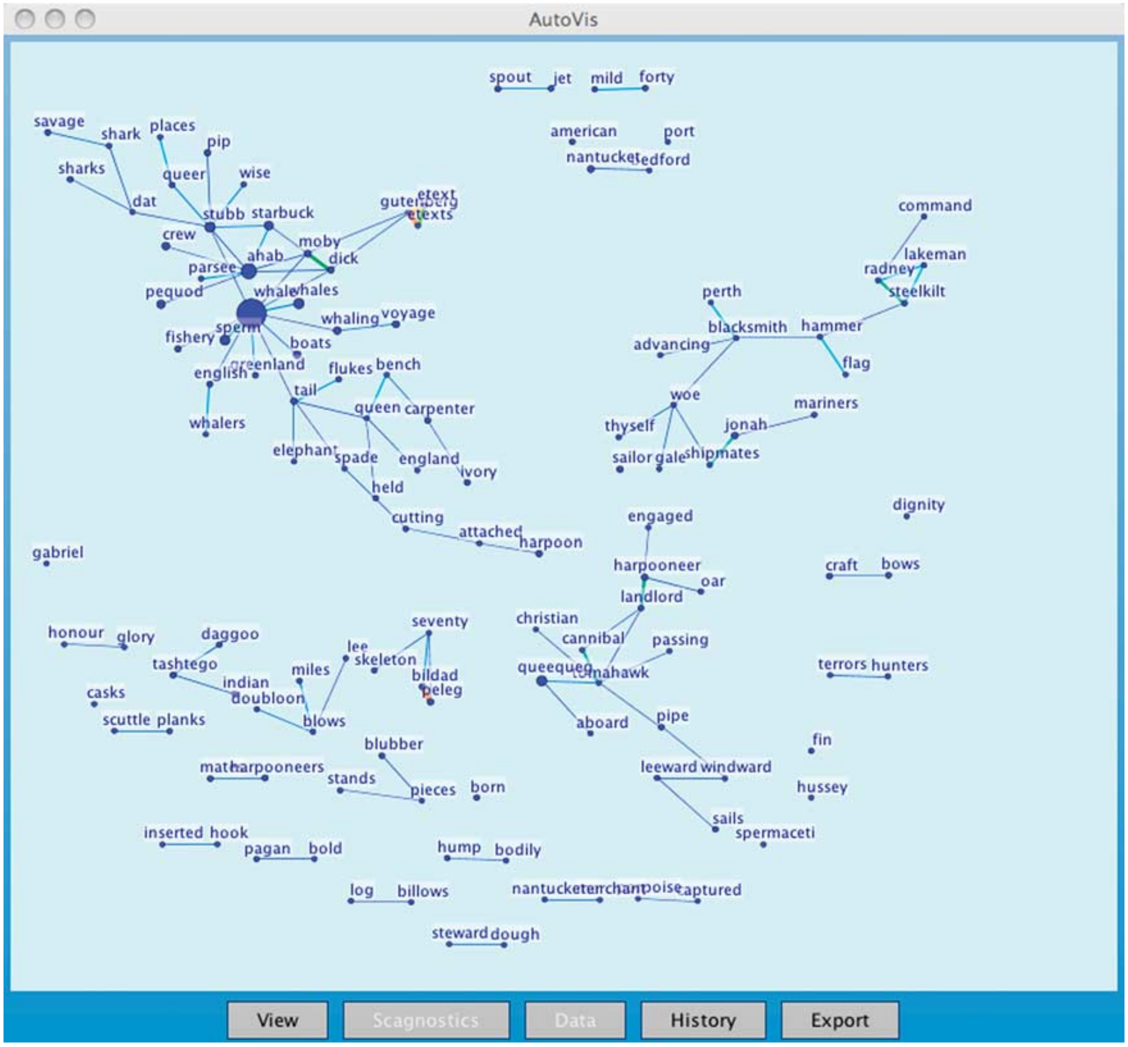


Figure 3: Text from Moby Dick. The edges in this graph were computed from co-occurrence of words in low-order *n*-grams. The size of the nodes is proportional to the frequency of the words in the text.

exponent near zero, we follow Tukey and select a *log* transformation. For values near one, we leave the data untransformed. Tukey calls this process *re-expression*. Most statisticians are familiar with a similar approach, called the Box-Cox transformation,[18] which was derived from Tukey's idea.

For categorical variables, we sort the category values and assign each category an integer value.

4. *Statistics*. Next, we compute statistical summaries. For continuous variables, we compute Epanechnikov-kernel-smoothed empirical histograms.[19] For categorical variables, we compute category frequencies. For pairs of continuous variables, we compute frequencies within hexagon-bins.[10,21] For continuous-categorical combinations, we compute Tukey letter values (quantiles) within category.[1] For pairs of categorical variables, we compute frequencies within rectangular bins.

   For free text, we compute a vertex-edge graph using *n*-grams and adjacencies. We use a force-directed graph layout[22] to compute coordinates of the vertices. For XML and other tree-based sources, we compute a tree by depth-first-search. For raster data, we compute color histograms and color-name distributions.
5. *Geometry*. We choose different geometric representations for different types of data. For Epanechnikov 1D densities, we use an *area* geometric element to highlight the density. For 1D frequency data, we choose a *bar* element. For 2D plots, we choose a *schema* (boxplot) element for continuous-categorical summaries, and a *tile* element for hexagon or rectangular bins. For

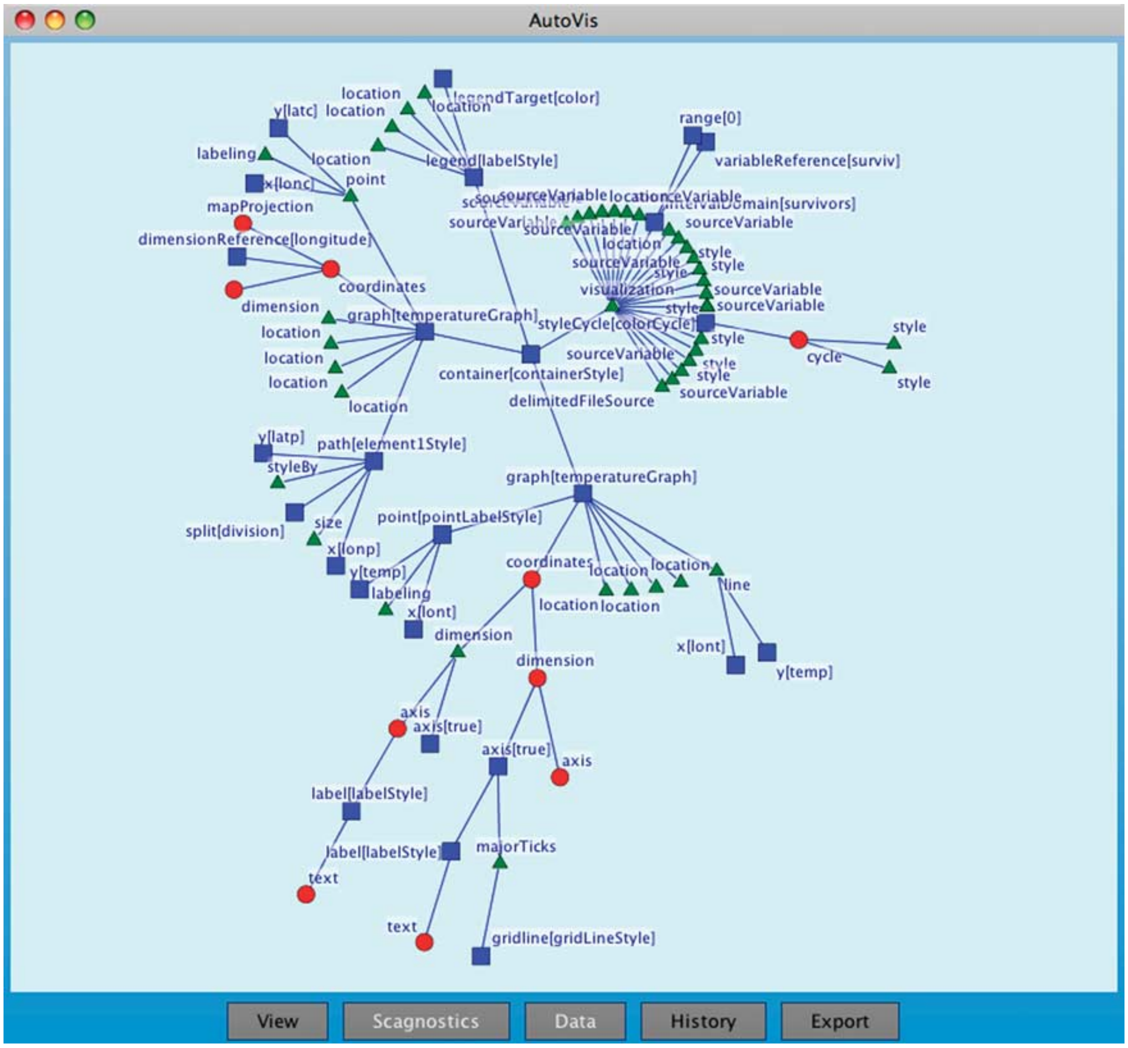


Figure 4: The grammar of graphics specification of Minard's graph of Napoleon's Russian campaign. This figure was created by dragging the visML specification (an XML schema) for the graph itself. The tree structure of the graph represents the hierarchy of the graph elements. Executing this tree in visML reproduces the Minard graph itself.

vertex-edge graphs, we use the *edge* element. These elements are defined in Appendix A.
None of these choices is necessary. We could assign different geometries to the same statistical graph elements. Nonetheless, we made our choices according to best practices among statisticians.[23,24]

6. *Coordinates*. Almost all the graphs are embedded in rectangular coordinates. To save display space, we use polar coordinates for tree layouts.[25]
7. *Aesthetics*. Almost all the graphs are rendered using *position* and *shape* aesthetic functions. We use color aesthetics sparingly in a few graphs in order to highlight noteworthy features. For example, we color tails of 1D densities red if they fall outside the region inside the Tukey *inner fences*:

$$q_{25} - 1.5(q_{75} - q_{25}) < R < q_{75} + 1.5(q_{75} - q_{25}) \quad (1)$$

where $q_{75} - q_{25}$ is the *interquartile range* or *H-spread*.[1] For normally distributed variables, using this criterion means that approximately 1 in 150 cases would be flagged as outliers.

Prioritizing

With more than a few variables to analyze, we face a priority problem. Presenting all 1D and 2D views on p variables requires $p(p+1)/2$ displays. This approach would use a prohibitive amount of display real estate, so we employ a strategy for finding both representative and unusual scatterplots.

Prioritizing is perhaps the most important task required for automatic visualization. By analogy, Google would be useless without the Page Rank statistic to order its search

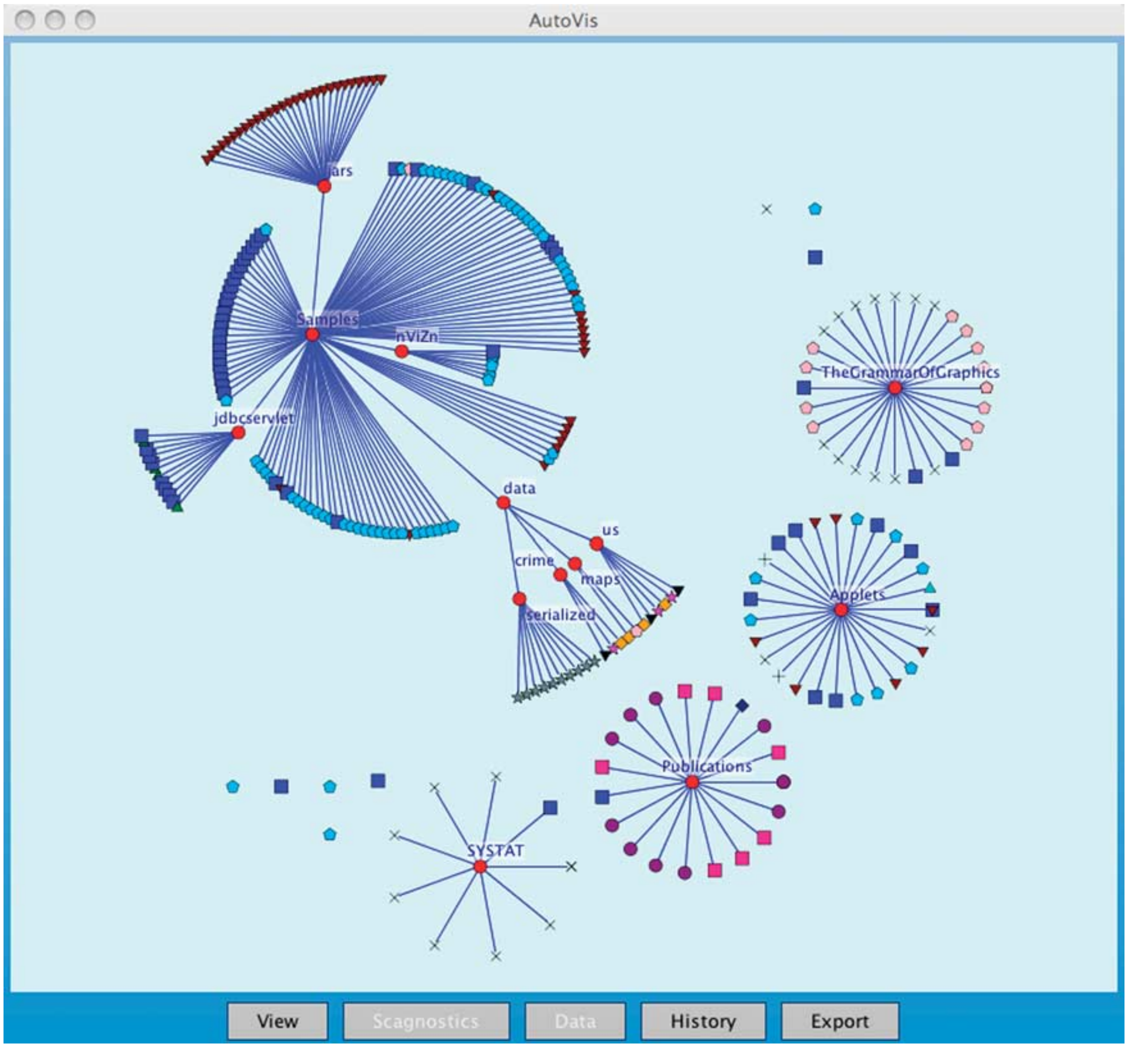


Figure 5: Website of the second author. The nodes are elements tagged in the HTML code for the site. The layout was produced by dragging the directory of the website into the AutoVis window.

results. We cannot use standard data mining methods (decision trees, support vector machines, neural networks, and so on) for this task because they presume a prediction or segmentation model. Automatic visualization needs to precede modeling.

Therefore, we need to operationalize the concept of 'interesting' in order to develop a mathematical model for discovering interesting patterns. For this, we use conventional statistical tests to filter out random relationships. We also use *scagnostics*[26] to identify interesting scatterplots. The scagnostic computations are outlined in Appendix B.

There are three kinds of displays we consider potentially interesting. The first involves *notables*. What are the significant relationships that should be considered before developing models? The second involves *exemplars*. What are the most typical joint distributions of points? The third involves *anomalies*. What are the most unusual or atypical distributions of points?

Notables

We draw on classical statistics to highlight noteworthy relationships. For continuous – continuous data, we compute a Spearman correlation coefficient. For continuous – categorical data, we compute a one-way analysis of variance. For categorical–categorical data, we compute a χ^2 measure of association. For each of these relations, we compute a conventional test of significance. We then sort all the P values and use the Benjamini–Hochberg formula to control the *false discovery error rate*[27]:

$$r_{BH} = \max\left\{0 \leqslant i \leqslant m : p_i \leqslant \alpha \frac{i}{m}\right\} \qquad (2)$$

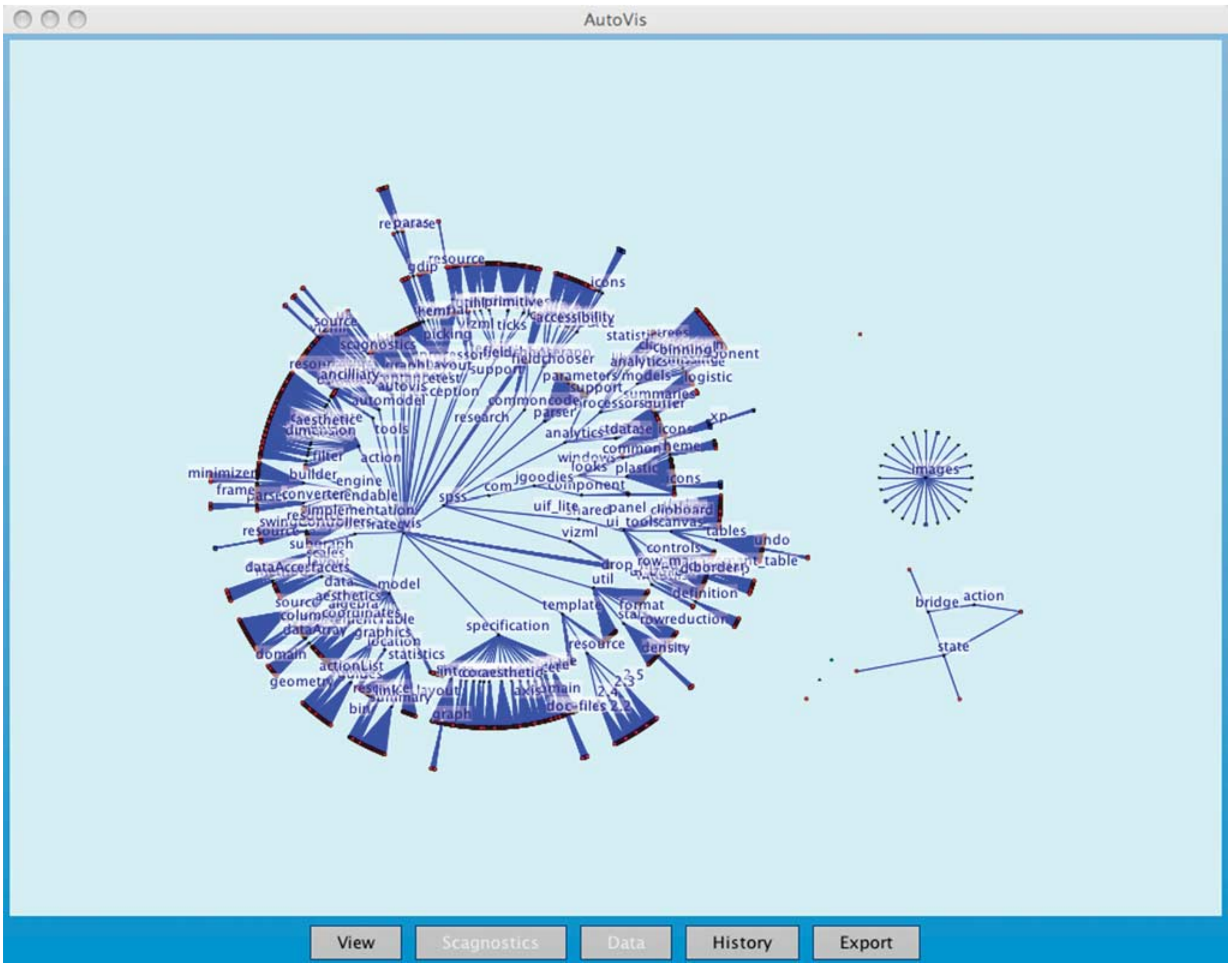


Figure 6: The AutoVis JAR file. AutoVis unzips the file automatically and analyzes its contents.

where $p_0 = 0$ and $\alpha = 0.01$ and m is the number of P values. We display all relationships for which $p_j \leqslant p_{r_{BH}}$.

Exemplars

We use an iterative hierarchical k-means cluster analysis to find exemplars.[28] This particular implementation of k-means includes a determination of the number of clusters through a sequential estimation technique. The resulting number of clusters depends on the distribution of the data. Further information on this algorithm and its performance is given at www.cs.uic.edu/~wilkinson/Applets/cluster.html.

The cluster analysis is computed on the scagnostics, not the original data. For 2D, there are nine scagnostics on $p(p-1)/2$ displays. Exemplars are computed by finding the scagnostic point nearest the centroid of each cluster. We then display the 2D plot corresponding to each scagnostics exemplar. We color the background of each plot a light green to signal that these are exemplars.

Anomalies

Finding anomalies requires a different approach. We could have computed Mahalanobis distances of every scagnostic point to each of the respective cluster centroids and used an F-test to determine outliers.[29] While Hartigan–Wong k-means clusters are guaranteed to be convex, they are not guaranteed to be multivariate Normal, even though the marginal null distributions of scagnostics are approximately Normal.[30] Consequently, we devised an alternative approach.

To locate anomalies, we compute a 9D geometric minimum spanning tree on the scagnostics. We then compute the distribution of edge lengths in the MST. We consider an outlier to be a vertex whose adjacent edges in

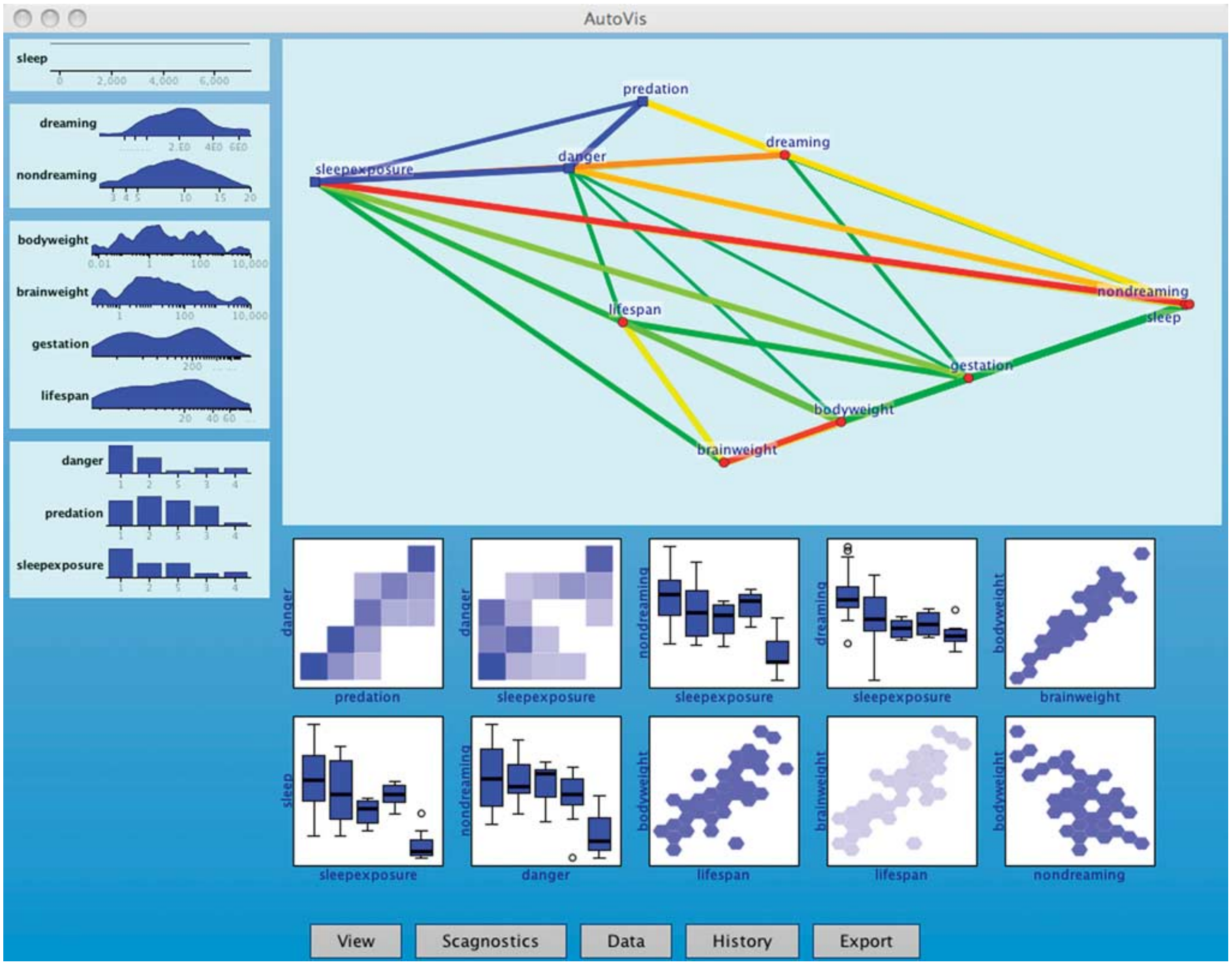


Figure 7: AutoVis representation of a statistical data set. The data are from a study of the relation between sleep habits of animals and their chance of being eaten by predators. The left column contains kernel densities of the continuous variables and bar charts for the categorical variables. The lower panels contain plots that AutoVis considers 'interesting.' The upper panel contains an association diagram based on a coefficient of association computed between pairs of variables.

the MST all have a length greater than F_{inner+}. Following Tukey,[1] we choose

$$F_{inner+} = q_{75} + 1.5(q_{75} - q_{25}) \qquad (3)$$

where q_{75} is the 75th percentile of the MST edge lengths and the expression in the parentheses is the *interquartile range* (Tukey's *hinge spread*) of the edge lengths. Tukey calls F_{inner+} the *upper inner fence* of a data batch. We display any 2D plot identified by this criterion. We color the background of each plot a light pink to signal that these are anomalies.

Summarizing

The tabular displays we have discussed so far are all 1D and 2D. We have so far devised methods for prioritizing the 2D displays by filtering out random and uninteresting relationships. Now we want to supplement these displays with a global summary that conveys relations among all the variables in a data source. Furthermore, in keeping with our statistical approach to prioritizing, we want this display to be unresponsive to random variation.

There are many off-the-shelf multivariate summary displays we could use for this purpose: parallel coordinates, glyphs (faces, rays, stars, and so on), scatterplot matrices, SVD-based projections (biplots, principal components, correspondence plots), and MDS-based projections. These methods suffer from various deficiencies: parallel coordinates and glyphs become messy for large n; scatterplot matrices become messy for large p; SVD does not handle nonlinearity; MDS is ill-suited for heterogeneous data. Furthermore, these standard methods do not address the randomness issue.

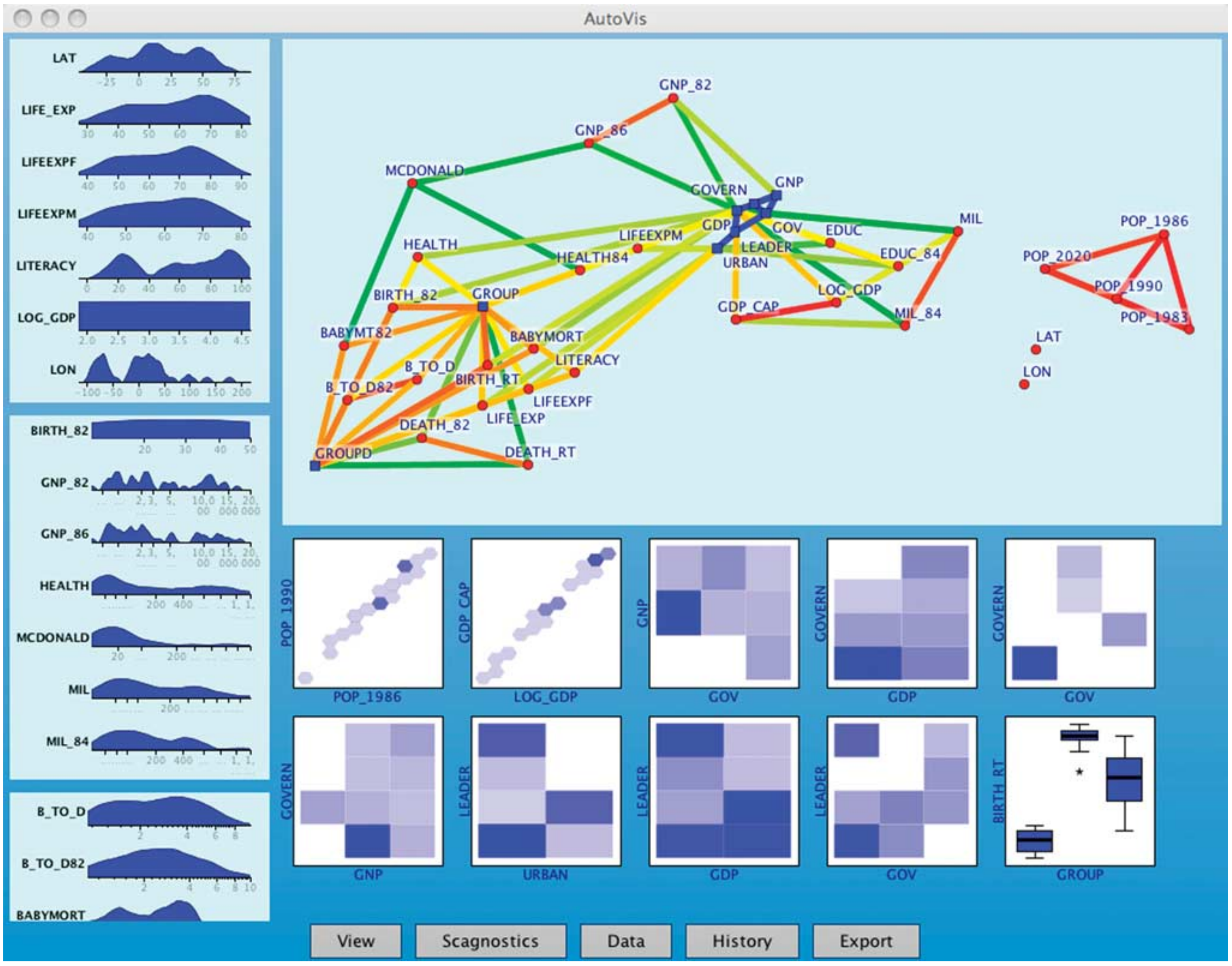


Figure 8: Social data from world countries data set. Variable clusters are discernible in the association map.

We decided to adopt an alternative approach that resembles some graph-theoretic displays used in computational biology. Namely, we use measures of similarity among variables to construct a weighted adjacency matrix and then embed the vertex-edge graph in a 2D display using the same force-directed layout algorithm we employ for text data. This approach deals flexibly with several problems.

First, we use several scaled association measures to address the heterogeneity problem. For continuous–continuous data, we compute the absolute value of a Spearman correlation coefficient. For continuous–categorical data, we compute an *eta* coefficient (square root of the ratio of between sum-of-squares to total sum-of-squares in a one-way analysis of variance). And for categorical – categorical data, we compute a *phi* coefficient (square root of χ^2/N). All three of these measures lie between 0 and 1.

Second, we color our edges to reflect goodness of fit. Red edges correspond to large association coefficients; blue correspond to small. If an edge is long and red, or if it is short and blue, we can see that the layout does not fit the data well.

Third, we set the thickness of our edges to reflect the strength of the relationship, but we threshold this setting. If an association is insignificant, the edge vanishes. We do this by computing a confidence interval on the parameter underlying the edge statistic, using the appropriate t or χ^2 distribution. Edges whose weights are within the confidence interval are deleted before layout. Vertices with no remaining adjacent edges are also deleted from the layout.

Examples

Figure 1 shows the opening window and user manual for AutoVis ('Drag files or text here'). There are no drop-down menus or modal tools. This software is minimalist.Figure 2 shows the history window one sees when pressing the

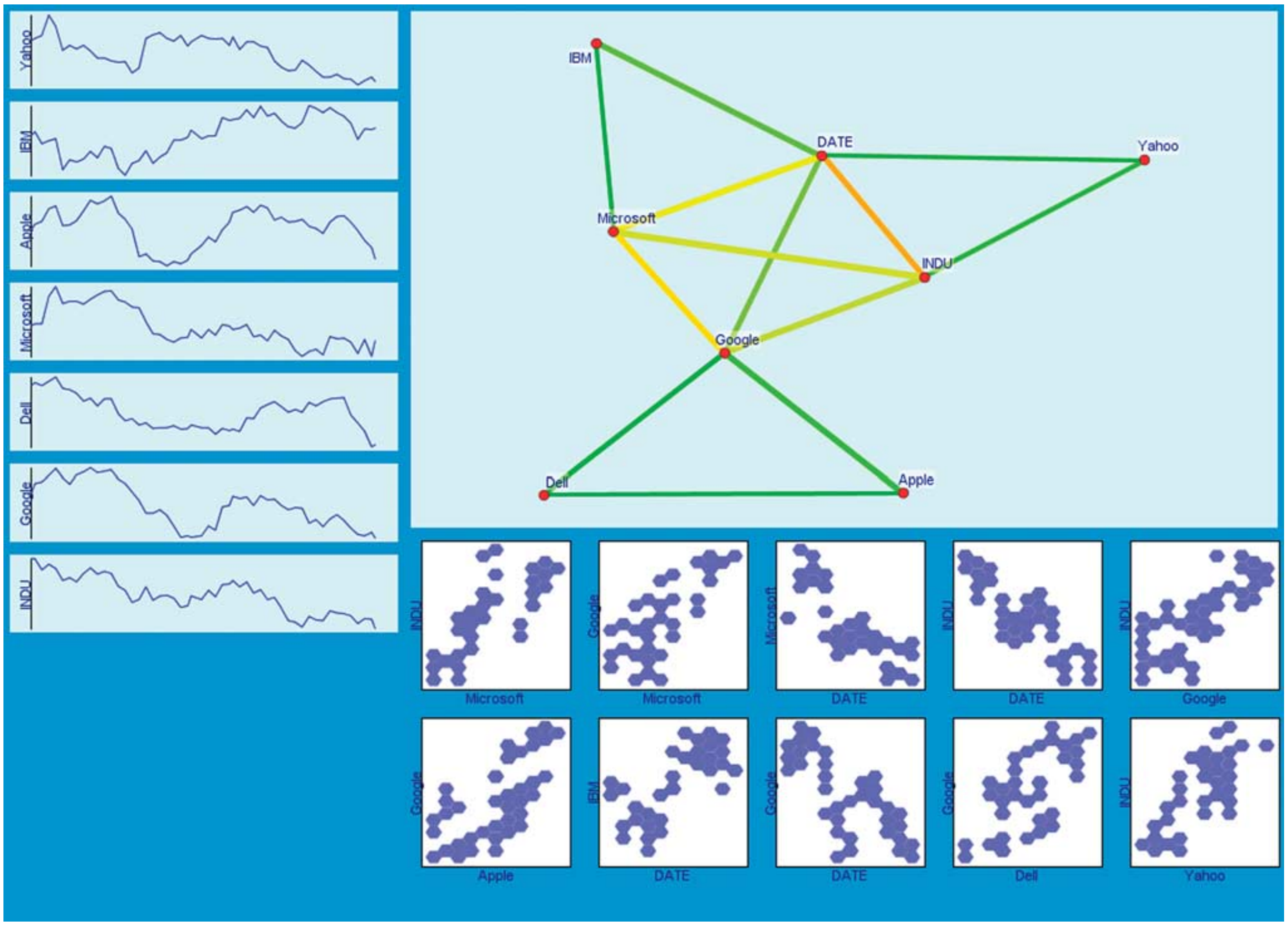


Figure 9: Time series data set. The multiple series represent stock prices for computer companies (Yahoo, IBM, Apple, Microsoft, Dell and Google). The Dow-Jones Industrial Average ($INDU) is also included.

History button. This view allows one to retrieve prior files to view them again. In this figure we show the files used in this paper plus a few additional files analyzed previously. The following sections illustrate the variety of data forms that AutoVis can recognize and display intelligently.

Networks (free text)

Figure 3 shows what happens when we drag a text file into the AutoVis window. We have selected text from *Moby Dick*. AutoVis recognizes that the file contains free text and accordingly computes an undirected graph based on an *n*-gram similarity measure. This measure counts the number of times two words appear jointly in a collection of text lists (sentences, paragraphs, or length-*n* sequences of words).[31,32] The full symmetric matrix of these counts defines a weighted adjacency matrix for a graph whose nodes are the words. AutoVis produces a planar layout of this graph using a force-directed algorithm.[22] It sizes nodes proportionally to the similarity measure. Not surprisingly, the word *whale* is the largest node in the graph.

Trees

AutoVis traverses a graph to discover if it is a tree. If so, it uses a polar layout algorithm to display the result. It is capable of recognizing a number of data formats that reduce to a tree structure.

XML

Figure 4 shows the result of dragging an XML resource into the AutoVis window. Upon recognizing that the file contains an XML tree, AutoVis parses the tree and computes a radial tree planar layout.[25]

The display contains the grammar-of-graphics vizML specification for Minard's famous map of Napoleon's Russian campaign.[33] Many of the grammar-of-graphics components are visible in this specification, as well as annotations such as axes and legends. Not surprisingly, the structure of this tree display is not unlike the

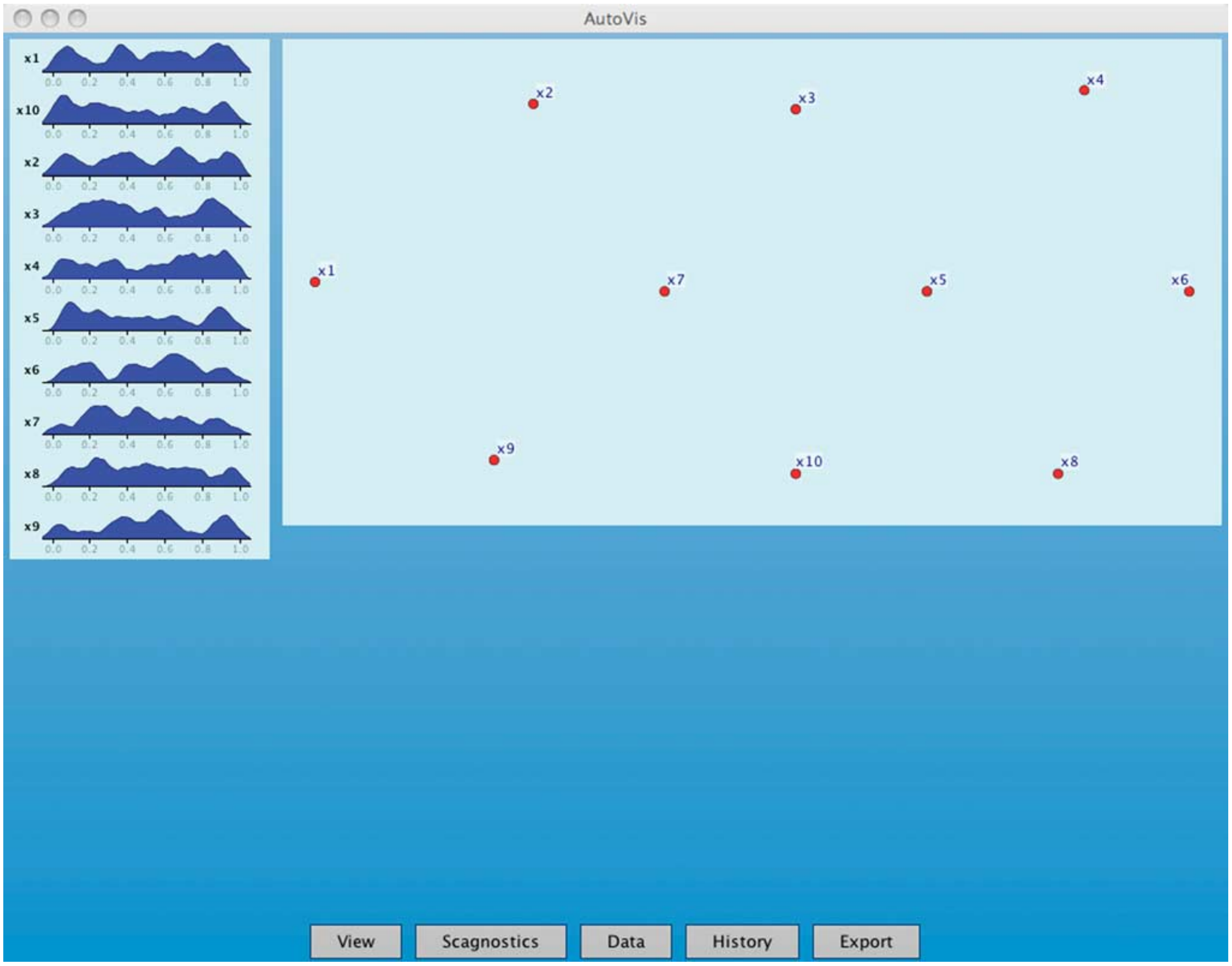


Figure 10: A random data set. AutoVis shows no associations because it finds none to be significant. It also finds no two-way plots to be 'interesting.' AutoVis is designed to protect users from making false conclusions while viewing data.

structure of Figure 1.2 in Wilkinson,[16] which shows the tree structure of a typical statistical graphic.

A website

Figure 5 shows the result of dragging a website into the AutoVis window. AutoVis computes the directory/resource tree of the site and uses the radial tree algorithm to compute a planar layout.

We have used the website of the second author (www.cs.uic.edu/~wilkinson/). There are clique nodes for *The Grammar of Graphics*, other publications, Java applets, SYSTAT and support files. This example reveals the label pruning that AutoVis does to avoid collisions. A popup tool allows one to hover over any node to examine its label interactively.

A Jar file

Figure 6 shows the result of dragging the Jar file for the AutoVis program itself. AutoVis unzips the file and analyzes the directory structure to lay out the contents. Note that the images are separated from the executables.

Tables

Upon recognizing that a data set is organized as a row/column table, AutoVis displays the entries through a number of informative statistical graphics. The graphs chosen for this purpose are derived from best practices among graphically oriented statisticians.[23,24] The specific graphs incorporated in a display depend on the form of the data.

Biological data

Figure 7 shows the result of dragging in a table containing biological measures. The data are from a study of the relation between sleep habits of animals and their chance of being eaten by predators.[34] On the left are kernel

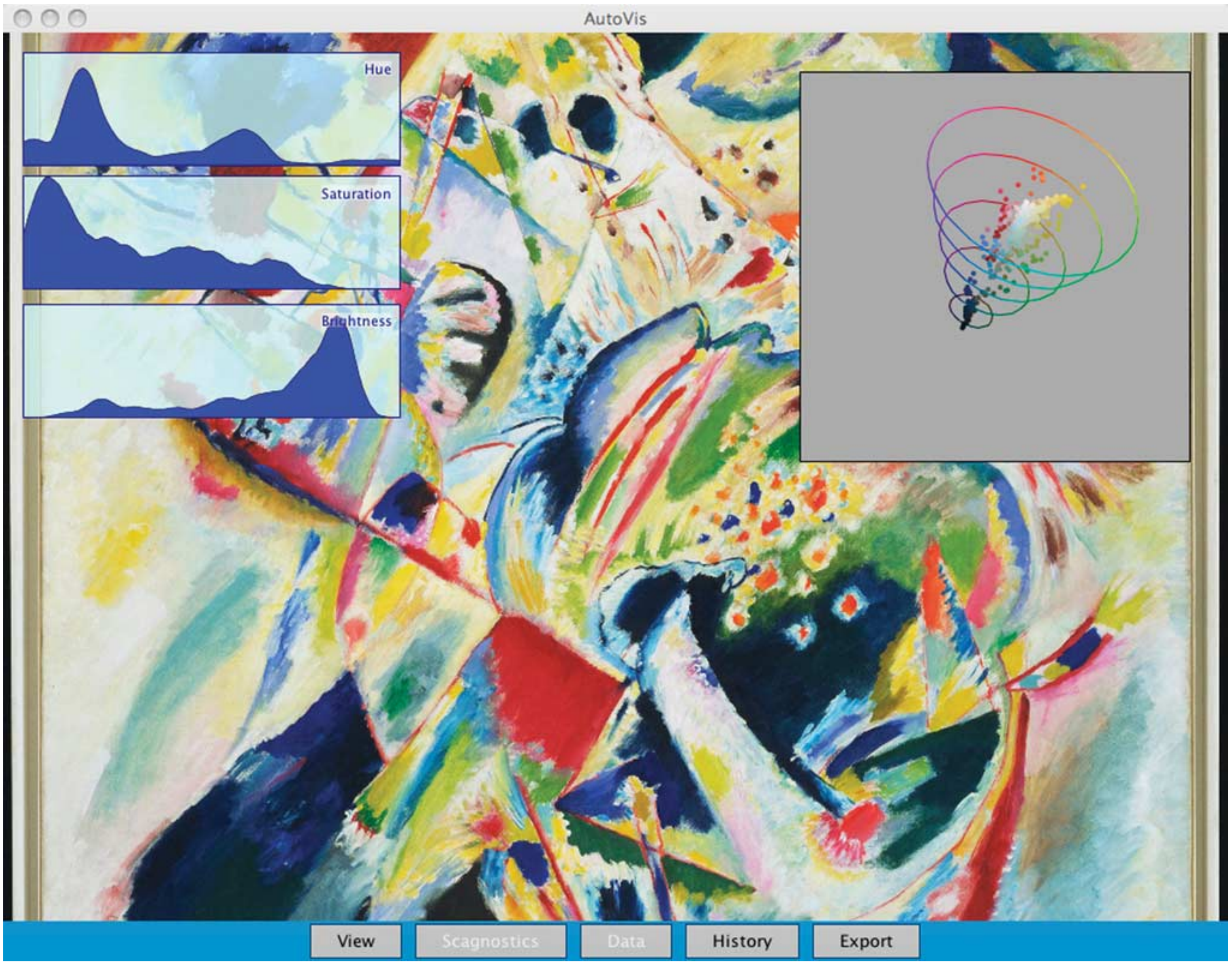


Figure 11: Kandinsky painting. The HSB color histogram is represented marginally on the left in three separate kernel densities, and jointly on the right in a rotatable 3D plot.

densities of the variables in the file. Notice that AutoVis has selected nonlinear scales for many of the continuous variables. AutoVis groups transformations into several panels. The top variable (sleep) is plotted on a linear scale; no transformation is needed. The next group of variables (dreaming and non-dreaming sleep) is plotted on a square-root scale. The four variables below this panel are plotted on a log scale. AutoVis gives us advice on transformations we should consider before doing statistical analyses. It decides on appropriate transformations by iteratively computing various transformations and examining the resulting histograms for skewness.

Directly below the kernel density panels are bar charts of three additional variables. Even though these variables are coded numerically in the file, AutoVis decided they are categorical and displayed them as bar charts.

The top right of the display contains an association diagram based on a robust measure of correlation between the variables. The layout algorithm is force-directed, similar to a multidimensional scaling. The strength of the association is represented by color (on a rainbow color scale). If the fit is good, then long edges should be red and short edges blue.

The remainder of the display shows an assortment of pairwise 2D plots. These have been selected by AutoVis using the various strength-of-association indices discussed above. The most strongly associated pairs of variables are displayed. Categorical displays are either box plots or tilings. Continuous displays are hexagon-binned scatterplots. The advantage of the tilings (rectangular or hexagonal) is that they work well with very large data sets, unlike ordinary scatterplots. These tilings are a form of joint 2D density estimates.

Social data

Figure 8 shows the result of dragging in a statistical data set with a larger number of variables. The data are excerpted

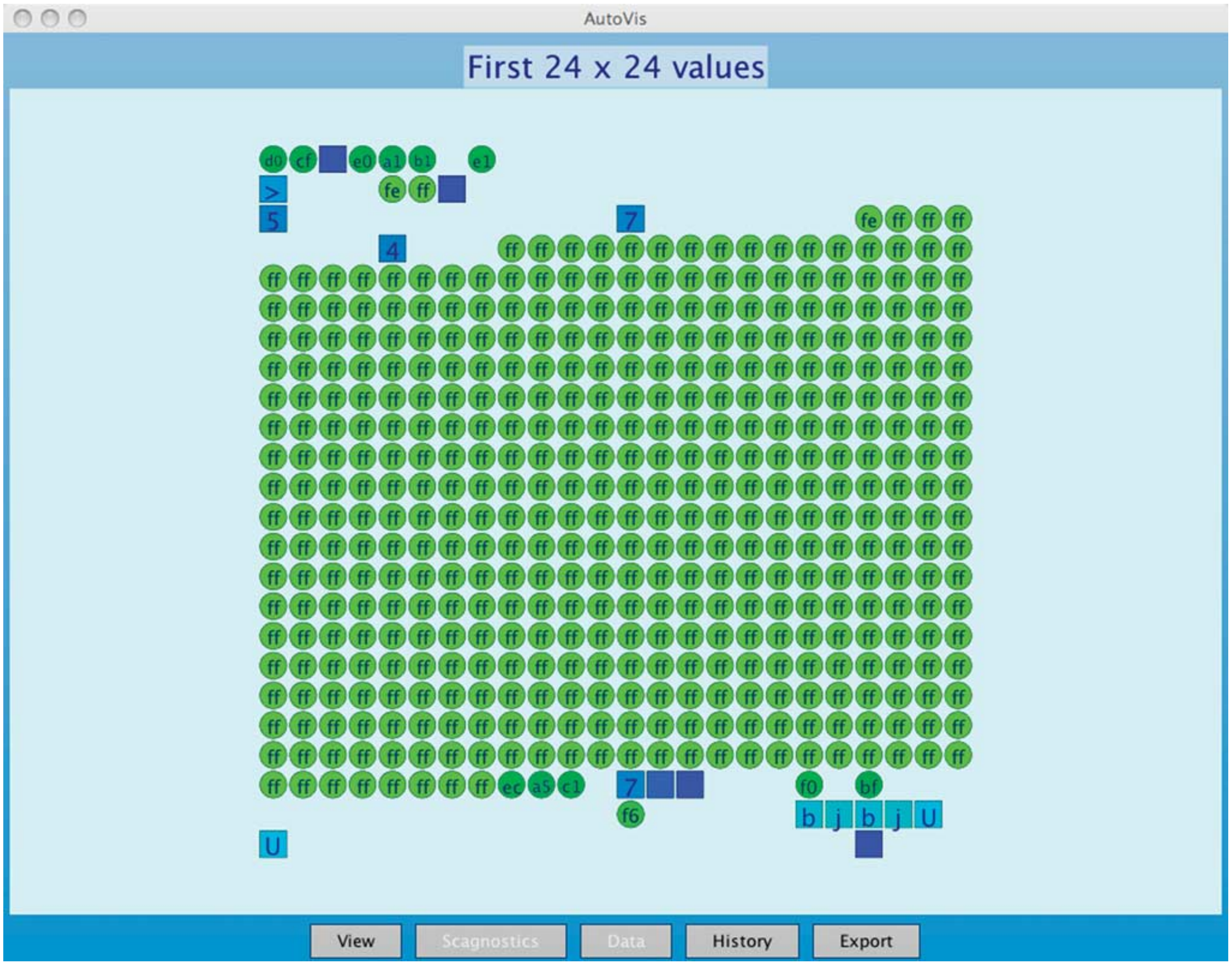


Figure 12: AutoVis throws up its hands and surrenders. The input is a Microsoft Word file with embedded macros and unusual text features not in the knowledge domain designed into AutoVis. In circumstances like this, AutoVis displays a 'core dump' of the contents instead of crashing.

from a UN databank of demographic variables cited in Wilkinson.[16] The most interesting aspect of this display is the way the association map groups the variables. We clearly see clusters of similar variables in this data set (infant mortality and birth rates, population, health statistics, economic and political measures, and death rates). The latitude and longitude variables, used for mapping the countries in this data set, sit appropriately by themselves. As with other displays, AutoVis transforms variables when it decides the transformations are helpful.

Time series data

Figure 9 contains a display of time series data. The data are stock series for several computer and software companies. After recognizing that the data comprise a time series, AutoVis switches the univariate displays on the left from kernel densities to line graphs. It also takes care not to represent time as the dependent variable on the vertical axis of any two-way plot.

Random data

Figure 10 shows the result of dragging in a uniform random data set the same size as the sleep data set. AutoVis shows no significant associations and finds no interesting two-way plots. It simply displays histograms (kernel densities) of the variables. No notables or anomalies are presented. This is one of the most important tests of the integrity of AutoVis. This demonstration underscores an important principle of visual analytics, one that is often neglected in practice: visualizations that display patterns in random data should be considered harmful because they can lead to false discoveries.

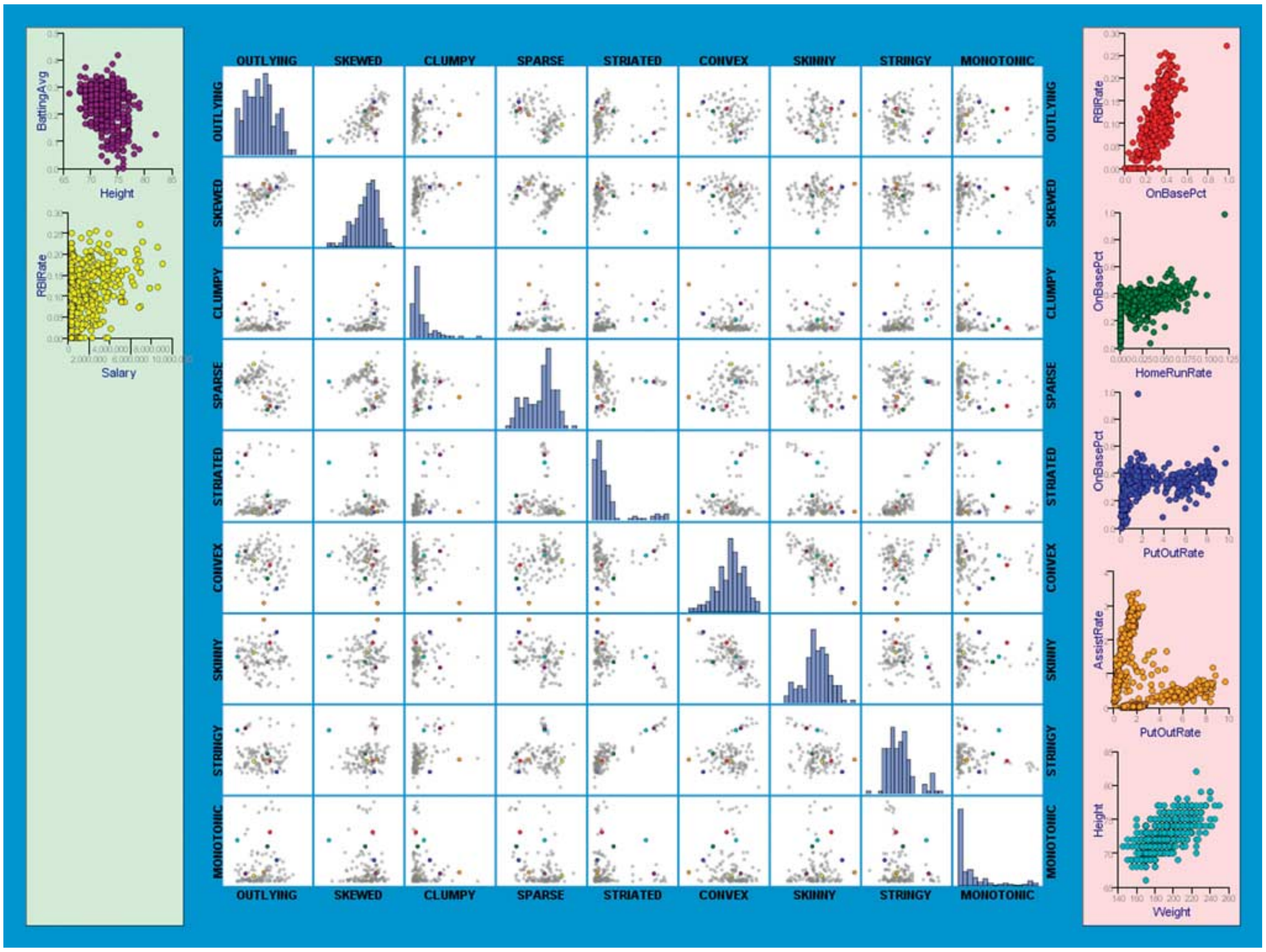


Figure 13: Scagnostics display panel of AutoVis. The data are statistics on 520 baseball players. The center panel shows the scatterplot matrix (SPLOM) of the scagnostics computed on these data. The left column shows two scatterplots considered by AutoVis to be exemplars for all the 120 possible scatterplots for the data set. The right column shows scatterplots that AutoVis considers to be anomalies.

Images

AutoVis recognizes a variety of graphic file formats. For image data, AutoVis computes the color distribution of the image itself.

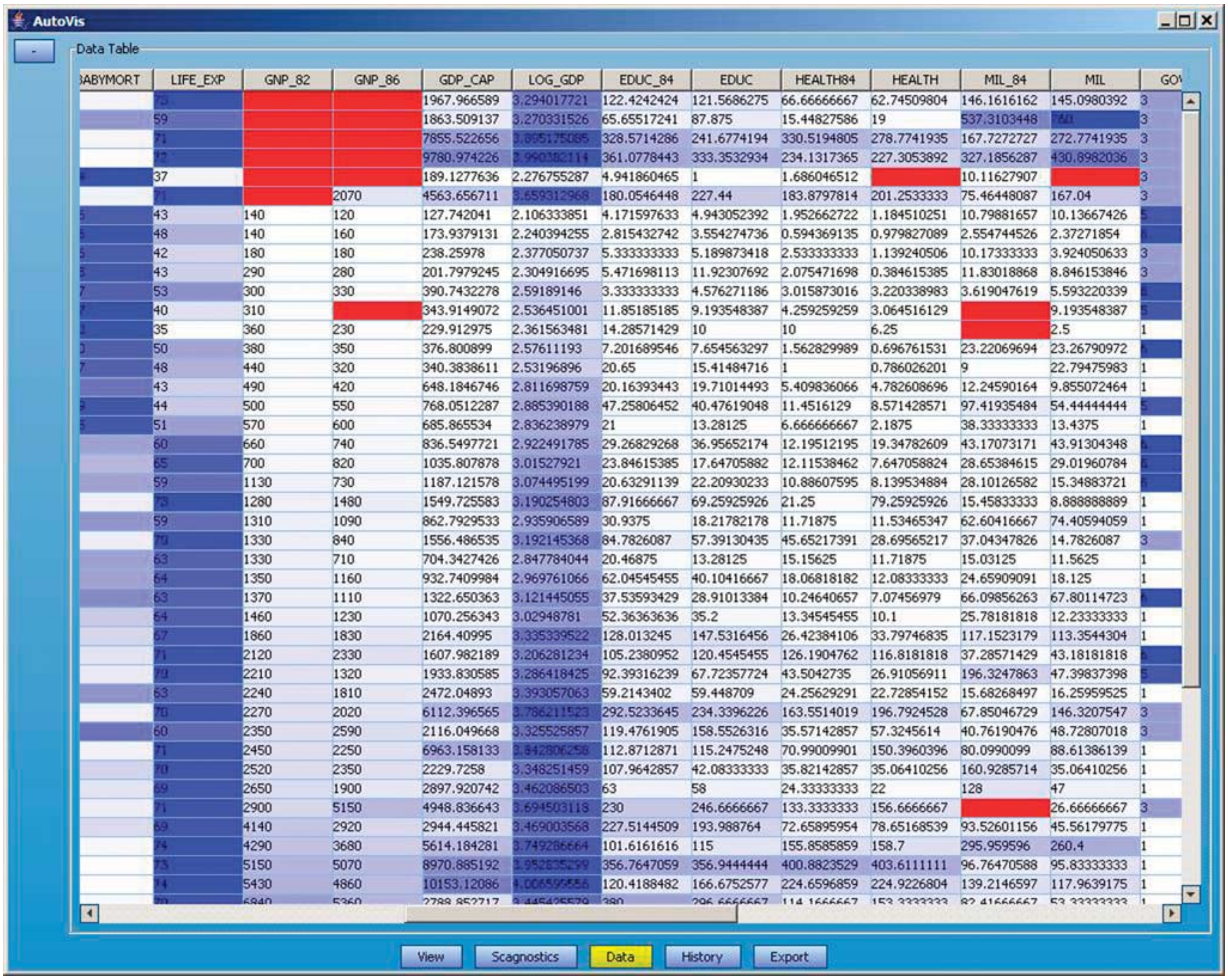
JPEG
Figure 11 shows the result of dragging a JPEG file into the AutoVis window. We have selected a painting of Wassily Kandinsky. On the left are kernel density histograms of the HSB components of the image. On the right is a rotatable (through mouse movements) 3D scatterplot of the HSB components.

Unrecognizable files

Figure 12 shows the result of dragging a file with a format unknown to AutoVis. In this case, AutoVis provides a color-highlighted hex dump of the file. The file in this example is a recent Microsoft Word document. We are working on decoding many variations of Word files, but parsing their structure from Java or other non-Microsoft environments is a non-trivial enterprise. This example illustrates the principle that the software should provide some representation of the data instead of crashing from failing to parse correctly.

Scagnostics

Figure 13 shows the scagnostics pane of AutoVis. The data are derived from statistics on 520 baseball players. The scatterplot matrix (SPLOM) in the center panel shows the nine scagnostics from Wilkinson *et al.*[26] computed on these data. On the left are two scatterplots that AutoVis considers to be exemplars. These plots were computed through a *k*-means cluster analysis of the scagnostics on all possible pairwise scatterplots. The algorithm continues to



BABYMORT	LIFE_EXP	GNP_82	GNP_86	GDP_CAP	LOG_GDP	EDUC_84	EDUC	HEALTH84	HEALTH	MIL_84	MIL	GOV
	75	🟥	🟥	1967.966589	3.294017721	122.4242424	121.5686275	66.66666667	62.74509804	146.1616162	145.0980392	3
	59	🟥	🟥	1863.509137	3.270331526	65.65517241	87.875	15.44827586	19	537.3103448	760	3
	71	🟥	🟥	7855.522656	3.895175085	328.5714286	241.6774194	330.5194805	278.7741935	167.7272727	272.7741935	3
	72	🟥	🟥	9780.974226	3.990382114	361.0778443	333.3532934	234.1317365	227.3053892	327.1856287	430.8982036	3
	37	🟥	🟥	189.1277636	2.276755287	4.941860465	1	1.686046512	🟥	10.11627907	🟥	3
	71	🟥	2070	4563.656711	3.659312968	180.0546448	227.44	183.8797814	201.2533333	75.46448087	167.04	3
	43	140	120	127.742041	2.106333851	4.171597633	4.943052392	1.952662722	1.184510251	10.79881657	10.13667426	5
	48	140	160	173.9379131	2.240394255	2.815432742	3.554274736	0.594369135	0.979827089	2.554744526	2.37271854	6
	42	180	180	238.25978	2.377050737	5.333333333	5.189873418	2.533333333	1.139240506	10.17333333	3.924050633	3
	43	290	280	201.7979245	2.304916695	5.471698113	11.92307692	2.075471698	0.384615385	11.83018868	8.846153846	3
	53	300	330	390.7432278	2.59189146	3.333333333	4.576271186	3.015873016	3.220338983	3.619047619	5.593220339	6
	40	310	🟥	343.9149072	2.536451001	11.85185185	9.193548387	4.259259259	3.064516129	🟥	9.193548387	5
	35	360	230	229.912975	2.361563481	14.28571429	10	10	6.25	🟥	2.5	1
	50	380	350	376.800899	2.57611193	7.201689546	7.654563297	1.562829989	0.696761531	23.22069694	23.26790972	6
	48	440	320	340.3838611	2.53196896	20.65	15.41484716	1	0.786026201	9	22.79475983	1
	43	490	420	648.1846746	2.811698759	20.16393443	19.71014493	5.409836066	4.782608696	12.24590164	9.855072464	1
	44	500	550	768.0512287	2.885390188	47.25806452	40.47619048	11.4516129	8.571428571	97.41935484	54.44444444	5
	51	570	600	685.865534	2.836238979	21	13.28125	6.666666667	2.1875	38.33333333	13.4375	1
	60	660	740	836.5497721	2.922491785	29.26829268	36.95652174	12.19512195	19.34782609	43.17073171	43.91304348	6
	65	700	820	1035.807878	3.01527921	23.84615385	17.64705882	12.11538462	7.647058824	28.65384615	29.01960784	6
	59	1130	730	1187.121578	3.074495199	20.63291139	22.20930233	10.88607595	8.139534884	28.10126582	15.34883721	6
	73	1280	1480	1549.725583	3.190254803	87.91666667	69.25925926	21.25	79.25925926	15.45833333	8.888888889	1
	59	1310	1090	862.7929533	2.935906589	30.9375	18.21782178	11.71875	11.53465347	62.60416667	74.40594059	1
	73	1330	840	1556.486535	3.192145368	84.7826087	57.39130435	45.65217391	28.69565217	37.04347826	14.7826087	3
	63	1330	710	704.3427426	2.847784044	20.46875	13.28125	15.15625	11.71875	15.03125	11.5625	1
	64	1350	1160	932.7409984	2.969761066	62.04545455	40.10416667	18.06818182	12.08333333	24.65909091	18.125	1
	63	1370	1110	1322.650363	3.121445055	37.53593429	28.91013384	10.24640657	7.07456979	66.09856263	67.80114723	6
	64	1460	1230	1070.256343	3.02948781	52.36363636	35.2	13.34545455	10.1	25.78181818	12.23333333	1
	67	1860	1830	2164.40995	3.335339522	128.013245	147.5316456	26.42384106	33.79746835	117.1523179	113.3544304	1
	71	2120	2330	1607.982189	3.206281234	105.2380952	120.4545455	126.1904762	116.8181818	37.28571429	43.18181818	6
	73	2210	1320	1933.830585	3.286418425	92.39316239	67.72357724	43.5042735	26.91056911	196.3247863	47.39837398	5
	63	2240	1810	2472.04893	3.393057063	59.2143402	59.448709	24.25629291	22.72854152	15.68268497	16.25959525	1
	70	2270	2020	6112.396565	3.786211523	292.5233645	234.3396226	163.5514019	196.7924528	67.85046729	146.3207547	3
	60	2350	2590	2116.049668	3.325525857	119.4761905	158.5526316	35.57142857	57.3245614	40.76190476	48.72807018	3
	71	2450	2250	6963.158133	3.842806258	112.8712871	115.2475248	70.99009901	150.3960396	80.0990099	88.61386139	1
	70	2520	2350	2229.7258	3.348251459	107.9642857	42.08333333	35.82142857	35.06410256	160.9285714	35.06410256	1
	69	2650	1900	2897.920742	3.462086503	63	58	24.33333333	22	128	47	1
	71	2900	5150	4948.836643	3.694503118	230	246.6666667	133.3333333	156.6666667	🟥	26.66666667	3
	69	4140	2920	2944.445821	3.469003568	227.5144509	193.988764	72.65895954	78.65168539	93.52601156	45.56179775	1
	74	4290	3680	5614.184281	3.749286664	101.6161616	115	155.8585859	158.7	295.959596	260.4	1
	73	5150	5070	8970.885192	3.952835299	356.7647059	356.9444444	400.8823529	403.6111111	96.76470588	95.83333333	1
	74	5430	4860	10153.12086	4.006595556	120.4188482	166.6752577	224.6596859	224.9226804	139.2146597	117.9639175	1

Figure 14: Data editor sorted within columns. The red cells denote missing values. The data are the world countries data set used in Figure 8. The blue highlighting denotes size of cell entries – darker blue corresponds to larger values.

split clusters until no statistically significant reduction in the within-cluster sum of squares is found. AutoVis then locates the scatterplots that are closest to the centroid of each of these clusters. From these, a data analyst can get an idea of the typical bivariate distributions without having to examine a huge number of pairwise plots.

On the right are five scatterplots that AutoVis considers to be anomalies. Three of these plots are singletons (the only examples of their shape). The plots listed here are the same anomalies shown in Wilkinson *et al.*[26] The statistical theory on which these distinctions are made is explained in Wilkinson and Wills.[30] It is based on a peeling of the minimum spanning tree on the scagnostics of all the plots.

Missing values

Missing values are highlighted in the data editor. AutoVis displays these values in red. It sorts the rows and columns so that similar patterns of missing values appear together in the sorted display.Figure 14 shows the AutoVis data editor containing the world countries data set used in Figure 8. Non-missing data are displayed in various saturations of blue. Larger values are highly saturated and smaller values are less saturated. For larger data sets, the editor reduces the size of the cells to accommodate more rows and columns.[35]

Conclusion

There are two unique premises to the research reported in this paper and inherent in the behavior of AutoVis. The first is that automatic visualization systems, and perhaps *all* visualization systems need to protect users from finding patterns in random data. The second is that automatic visualization systems must be designed to follow the process that expert analysts use in examining data.

Protecting users from false conclusions based on random data is paramount. User judgments of meaningfulness cannot substitute for mathematical analysis. People can be fooled into thinking relatively simple programs are intelligent.[36] Even experts can be fooled into finding patterns in random data.[37–39] This is why experts rely on statistical procedures based on probability theory to protect themselves from false conclusions. Expert visualization systems must do the same.

Second, expert visualization systems need to follow the same rules that expert analysts follow in practice. It would be relatively simple to short cut this effort by developing a system that achieves the surface appearance of AutoVis through a standard visualization library and some basic statistical graphics. By hard-wiring graph types to data types, we could produce visualizations for a variety of data sets. A novice viewer might think that such a system was flexible and intelligent.

We are not interested in making a popular tool, however. We are interested in understanding how to model the process underlying what an expert analyst does when screening data. AutoVis is a research platform for this effort. It is not designed to produce captivating visualizations. It is designed to mimic the steps expert analysts follow when they first examine data and to display the types of statistical graphics experts have chosen as best-of-breed for this purpose.[23,24,40]

We intend to evaluate AutoVis in a user experiment. Such an experiment will not be easy to design, however. First of all, we need to develop several classes of outcome measures. One class of measures involves user satisfaction with the software: do novice and expert users alike feel that the software facilitates their exploration and analysis? Another class involves user performance: does using AutoVis produce a more valid inference (as measured by integrated mean-square error on replication samples from known populations) than using a statistical package or interactive visual analytic system? Another class involves appropriateness of the visualizations: do the visualizations produced by AutoVis correspond to those produced by expert statisticians examining the same data set?

We are not ready to evaluate the current AutoVis interface, however. We need to do more work on broadening the inferential capabilities of the AutoVis engine before settling on a production user interface. Specifically, we need to handle more data structures to make the application useful in real settings. We need to develop spatial mappings of non-numerical data (text, video, audio) in order to exploit the pattern-recognition capabilities of Scagnostics. And we need to implement gestures that will make AutoVis easy to use in a web environment.

We expect to refine our model as we get more experience testing AutoVis on real data sets. Mackinlay posed the original problem for relational data. Our research is heading toward realizing Mackinlay's goals for almost any type of data.

Acknowledgements

Roger Dubbs and Dan Rope provided valuable suggestions. The work on this paper was supported in part by the corresponding authors' National Science Foundation grant DMS-FODAVA 0808860. AutoVis can be explored at www.cs.uic.edu/~wilkinson/Applets/autovis.html.

References

1 Tukey, J.W. (1977). *Exploratory Data Analysis*. Reading, MA: Addison-Wesley Publishing Company.
2 Friedman, J.H. and Stuetzle, W. (2002). John W. Tukey's work on interactive graphics. *The Annals of Statistics* 30: 1629–1639.
3 Huber, P.J. (2002). John W. Tukey's contributions to robust statistics. *The Annals of Statistics* 30(6): 1640–1648.
4 Mackinlay, J. (1986). Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics (TOG)* 5(2): 110–141.
5 Roth, S.F., Kolojechick, J., Mattis, J. and Goldstein, J. (1994). Interactive graphic design using automatic presentation knowledge. In: C. Plaisant (ed.) *CHI '94: Conference Companion on Human Factors in Computing Systems*, New York, NY, USA. New York: ACM Press, p. 207.
6 Roth, S., Chuah, M., Kerpedjiev, S., Kolojejchick, J. and Lucas, P. (1997). Towards an information visualization workspace: Combining multiple means of expression. *Human–Computer Interaction Journal* 12: 131–185.
7 Petajan, E.D., Jean, Y.D., Lieuwen, D. and Anupam, V. (1997). Dataspace: An automated visualization system for large databases. In: G.G. Grinstein and R.F. Erbacher (eds.) *Proceedings of the SPIE, Visual Data Exploration and Analysis IV*. Silver Spring, MD: IEEE Computer Society Press, pp. 89–98.
8 Casner, S.M. (1991). Task-analytic approach to the automated design of graphic presentations. *ACM Transactions on Graphics* 10(2): 111–151.
9 Iizuka, Y., Shiohara, H., Iizuka, T. and Isobe, S. (1998). Automatic visualization method for visual data mining. In: X. Wu, K. Ramamohanarao, K.B. Korb (eds.) *PAKDD '98: Proceedings of the Second Pacific-Asia Conference on Research and Development in Knowledge Discovery and Data Mining*, London, UK. Berlin: Springer-Verlag, pp. 173–185.
10 Chuah, M.C. (2000). *AVID: Automatic visualization interface designer*. PhD thesis, School of Computer Science, Carnegie Mellon University.
11 Senay, H. and Ignatius, E. (1994). A knowledge-based system for visualization design. *IEEE Computer Graphics Applications* 14(6): 36–47.
12 Keim, D.A. (2002). Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics* 8(1): 1–8.
13 Cruz-Neira, C., Sandin, D.J., DeFanti, T.A., Kenyon, R.V. and Hart, J.C. (1992). The cave: Audio visual experience automatic virtual environment. *Communications of the ACM* 35(6): 64–72.
14 Jerding, D.F. and Stasko, J.T. (1998). The information mural: A technique for displaying and navigating large information spaces. *IEEE Transactions on Visualization and Computer Graphics* 4(3): 257–271.
15 Czernuszenko, M., Pape, D., Sandin, D., DeFanti, T., Dawe, G.L. and Brown, M.D. (1997). The immersadesk and infinity wall projection-based virtual reality displays. *SIGGRAPH Comput. Graph.* 31(2): 46–49.
16 Wilkinson, L. (2005). *The Grammar of Graphics*, 2nd edn. New York: Springer-Verlag.
17 Tukey, J.W. (1957). On the comparative anatomy of transformations. *Annals of Mathematical Statistics* 28: 602–632.
18 Box, G.E.P. and Cox, D.R. (1964). An analysis of transformations. *Journal of the Royal Statistical Society, Series B* 26: 211–252.

19 Silverman, B. (1986). *Density Estimation for Statistics and Data Analysis*. New York: Chapman & Hall.

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

21 Carr, D.B. (1993). Looking at large data sets using binned data plots. In: A. Buja and P. Tukey (eds.) *Computing and Graphics in Statistics*. New York: Springer-Verlag, pp. 7–39.

22 Fruchterman, T.W.J. and Reingold, E.M. (1991). Graph drawing by force-directed placement. *Software – Practice and Experience* 21(11): 1129–1164.

23 Chambers, J.M., Cleveland, W.S., Kleiner, B. and Tukey, P.A. (1983). *Graphical Methods for Data Analysis*. New York: Chapman & Hall.

24 Cleveland, W.S. (1985). *The Elements of Graphing Data*. Summit, NJ: Hobart Press.

25 Wills, G.J. (1999). NicheWorks – Interactive visualization of very large graphs. *Journal of Computational and Graphical Statistics* 8(2): 190–212.

26 Wilkinson, L., Anand, A. and Grossman, R. (2006). High-dimensional visual analytics: Interactive exploration guided by pairwise views of point distributions. *IEEE Transactions on Visualization and Computer Graphics* 12(6): 1363–1372.

27 Benjamini, Y. and Hochberg, Y. (1995). Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society, Series B* 57: 289–300.

28 Hartigan, J.A. and Wong, M. (1979). A *k*-means clustering algorithm. *Applied Statistics* 28: 100–108.

29 Rao, C.R. (1973). *Linear Statistical Inference and its Applications*, 2nd edn. New York: John Wiley & Sons.

30 Wilkinson, L. and Wills, G. (2008). Scagnostics distributions. *Journal of Computational and Graphical Statistics*, in press.

31 Rosenberg, S. and Jones, R. (1972). A method for investigating and representing a person's implicit theory of personality: Theodore dreiser's view of people. *Journal of Personality and Social Psychology* 22: 372–386.

32 Damashek, M. (1995). Gauging similarity with n-grams: Language-independent categorization of text. *Science* 267: 843–848.

33 Tufte, E.R. (2002). *The Visual Display of Quantitative Information*, 2nd edn. Cheshire, CT: Graphics Press.

34 Allison, T. and Cicchetti, D. (1976). Sleep in mammals: Ecological and constitutional correlates. *Science* 194: 732–734.

35 Rao, R. and Card, S.K. (1994). The table lens: Merging graphical and symbolic representations in an interactive focus context visualization for tabular information. In: B. Adelson, S. Dumais, and J. Olson (eds.) *Proceedings of the ACM Conference on Human Factors in Computing Systems CHI*. New York: ACM.

36 Weizenbaum, J. (1966). Eliza – A computer program for the study of natural language communication between man and machine. *Communications of the ACM* 9(1): 36–45.

37 Payne, J.L. (1974). Fishing expedition probability: The statistics of *post hoc* hypothesizing. *Polity* 7(1): 130–138.

38 Tversky, A. and Kahneman, D. (1983). Extension versus intuitive reasoning: The conjunction fallacy in probability judgment. *Psychological Review* 90: 293–315.

39 Gilovich, T., Vallone, R. and Tversky, A. (1985). The hot hand in basketball: On the misperception of random sequences. *Cognitive Psychology* 17: 295–314.

40 Cleveland, W.S. and McGill, R. (1984). Graphical perception: Theory, experimentation, and application to the development of graphical methods. *Journal of the American Statistical Association* 79: 531–554.

41 Stevens, S.S. (1946). On the theory of scales of measurement. *Science* 103: 677–680.

42 Velleman, P. and Wilkinson, L. (1993). Nominal, ordinal, interval, and ratio typologies are misleading for classifying statistical methodology. *The American Statistician* 47: 65–72.

43 Taylor, B.N. (1997). The international system of units (SI). *Special Publication 330*, NIST, Washington, DC. pp. 171–182.

44 Bertin, J. (1967). *Sèmiologie Graphique*. Paris: Editions Gauthier Villars.

45 Kruskal, J.B. (1956). On the shortest spanning subtree of a graph and the travelling salesman problem. *Proceedings of the American Mathematical Society* 7: 48–50.

46 Jaromczyk, J.W. and Toussaint, G.T. (1992). Relative neighborhood graphs and their relatives. *Proceedings of the IEEE* 80: 1502–1517.

47 Edelsbrunner, H., Kirkpatrick, D.G. and Seidel, R. (1983). On the shape of a set of points in the plane. *IEEE Transactions on Information Theory* 29: 551–559.

48 Marchette, D. (2004). *Random Graphs for Statistical Pattern Recognition*. New York: John Wiley & Sons.

49 Freedman, D.A. (2001). Ecological inference and the ecological fallacy. In: N.J. Smelser and P.B. Baltes (eds.) *International Encyclopedia of the Social and Behavioral Sciences*. Oxford: Pergamon Press, pp. 4027–4030.

50 Barnett, V. and Lewis, T. (1994). *Outliers in Statistical Data*. New York: John Wiley & Sons.

51 Tukey, J.W. (1974). Mathematics and the picturing of data. *Proceedings of the International Congress of Mathematicians*. Vancouver, Canada. Canadian Mathematical Congress, pp. 523–531.

52 Dixon, W.J. and Tukey, J.W. (1968). Approximate behavior of the distribution of winsorized *t*. *Technometrics* 10: 83–98.

53 Hartigan, J.A. and Mohanty, S. (1992). The runt test for multimodality. *Journal of Classification* 9: 63–70.

54 Gower, J.C. and Ross, G.J.S. (1969). Minimal spanning trees and single linkage cluster analysis. *Applied Statistics* 18: 54–64.

55 Stuetzle, W. (2003). Estimating the cluster tree of a density by analyzing the minimal spanning tree of a sample. *Journal of Classification* 20: 25–47.

Appendix A. The Grammar of Graphics

The Grammar of Graphics, or GoG,[16] denotes a system with seven orthogonal components. By *orthogonal*, we mean there are seven component classes, each containing one or more methods (functions) as elements, and all tuples in the seven-fold product of these sets of functions produce meaningful graphs. A consequence of this orthogonality is a high degree of expressiveness: we can produce a huge variety of graphical forms or chart types in such a system. In fact, it is claimed that virtually the entire corpus of known charts can be generated by this relatively parsimonious system, and perhaps a great number of meaningful but undiscovered chart types as well. For the Java platform used in AutoVis (called nViZn), the product set of the seven function classes generates hundreds of thousands of different graphs.

The second principal claim of GoG is that this system describes the meaning of what we do when we construct statistical graphs, charts and visualizations. It is more than a taxonomy. It is a computational system based on the underlying mathematics of representing functions of data.

Figure A1 shows a *data flow* diagram that contains the seven GoG classes. This data flow is a chain that describes the sequence of mappings needed to produce a statistical graphic from a set of data. The first class (Variables) maps data to an object called a *varset* (a set of variables). The next two classes (Algebra, Scales) are transformations on varsets. The next class (Statistics) takes a varset and creates a statistical graph (a statistical summary). The next class (Geometry) maps a statistical graph to a geometric graph. The next (Coordinates) embeds a graph in a coordinate

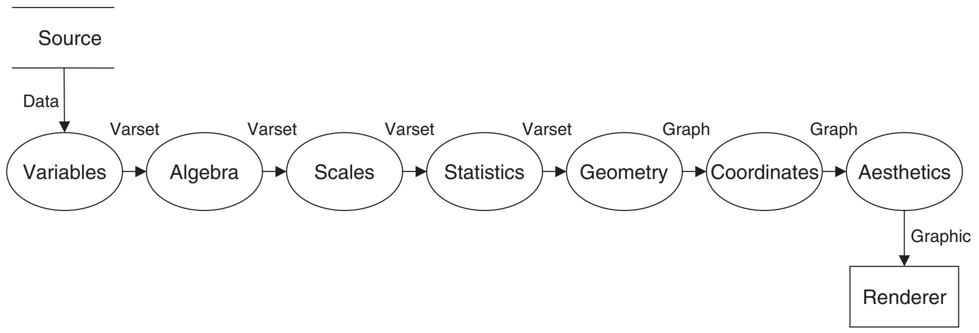


Figure A1: The grammar of graphics data flow.

space. And the last class (Aesthetics) maps a graph to a visible or perceivable display called a graphic.

The data flow architecture implies that the subtasks needed to produce a graphic from data must be done in this specified order. Changes to this ordering can produce meaningless graphics. For example, if we compute certain statistics on variables (for example, sums) before scaling them (for example, log scales), we can produce statistically meaningless results because the log of a sum is not the sum of the logs.

The data flow in has many paths through it. We can choose different algebraic designs (factorial, nested, and so on), scales (log, probability, and so on), statistical methods (means, medians, modes, smoothers, and so on), geometric objects (points, lines, bars, and so on), coordinate systems (rectangular, polar, and so on), and aesthetics (size, shape, color, and so on). These paths reveal the richness of the system. We now summarize the seven GoG classes.

Variables

We begin with data. We assume the data that we wish to graph are organized in one or more tables. The column(s) of each table represent a set of fields, each field containing a set of measurements or attributes. The row(s) of this table represent a set of logical records, each record containing the measurements of an object on each field. Usually, a relational database management system (RDBMS) produces such a table from organized queries specified in Structured Query Language (SQL) or another relational language. Other data sources (object, streaming, and so on) can be mapped to tables through similar methods.

Our first step is to convert a table of data to a *varset*. A varset is a set of one or more variables. While a column of a table of data might superficially be considered to be a variable, there are differences. A variable is both more general (in regard to generalizability across samples) and more specific (in regard to data typing and other constraints) than a column of data. First, we define a variable, then a varset.

Variable

A *variable* X is a mapping $f : O \to V$, which we consider as a triple:

$$X = [O, V, f]$$

The domain O is a set of objects.
The codomain V is a set of values.
The function f assigns to each element of O an element in V.

The image of O under f contains the *values* of X. We denote a possible value as x, where $x \in V$. We denote a value of an object as $X(o)$, where $o \in O$. A variable is *continuous* if V is an interval. A variable is *categorical* if V is a finite subset of the integers (or there exists an injective map from V to a finite subset of the integers).

Variables may be multidimensional. $\boldsymbol{X}$ is a p-dimensional variable made up of p one-dimensional variables:

$$\begin{aligned}\boldsymbol{X} &= (X_1, \ldots, X_p)\\ &= [O, V_i, f], \quad i = 1, \ldots, p\\ &= [O, \boldsymbol{V}, f]\end{aligned}$$

The element $\boldsymbol{x} = (x_1, \ldots, x_p)$, $\boldsymbol{x} \in \boldsymbol{V}$, is a p-dimensional value of $\boldsymbol{X}$. We use multidimensional variables in *multivariate analysis*.

Varset

We call the triple

$$\mathrm{X} = [V, \widetilde{O}, f]$$

a *varset*. The word *varset* stands for *variable set*. If X is multidimensional, we use boldface **X**. A varset inverts the mapping used for variables. That is,

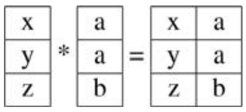
The domain V is a set of values.
The codomain $\widetilde{O}$ is a set of all possible ordered lists of objects.
The function f assigns to each element of V an element in $\widetilde{O}$.

We invert the mapping customarily used for variables in order to simplify the definitions of graphics algebra operations on varsets. In doing so, we also replace the variable's set of objects with the varset's set of ordered lists. We use lists in the codomain because it is possible for a value to be mapped to an object more than once (as in repeated measurements).

Algebra

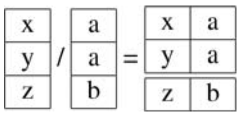
Given one or more varsets, we now need to operate on them to produce combinations of variables. A typical scatterplot of a variable X against a variable Y, for example, is built from tuples (x_i, y_i) that are elements in a set product. We use graphics algebra on values stored in varsets to make these tuples. There are three binary operators in this algebra: *cross*, *nest* and *blend*.

Cross $(*)$
Cross joins the left argument with the right to produce a set of tuples stored in the multiple columns of the new varset:



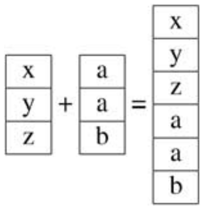
The resulting set of tuples is a subset of the product of the domains of the two varsets. The domain of a varset produced by a cross is the product of the separate domains. One may think of a cross as a horizontal concatenation of the table representation of two varsets, assuming the rows of each varset are equivalent and in the same order.

Nest $(/)$
Nest partitions the left argument using the values in the right:



Although it is not required in the definition, we assume that the nesting varset on the right is categorical. The name *nest* comes from design-of-experiments terminology. We often use the word *within* to describe its effect. For example, if we assess schools and teachers in a district, then *teachers within schools* specifies that teachers are nested within schools. Assuming each teacher in the district teaches at only one school, we would conclude that if our data contain two teachers with the same name at different schools, they are different people.

Blend $(+)$
Blend produces a union of varsets:



Blend is defined only if the order of the tuples (number of columns) in the left and right varsets is the same. Furthermore, we need to restrict blend to varsets with composable domains. It would make little sense to blend Age and Weight, much less Name and Height. The Scales class, in the next section, throws an exception if we attempt to blend varsets across different types of scales.

Scales

Before we compute summaries (totals, means, smoothers, and so on) and represent these summaries using geometric objects (points, lines, and so on), we must scale our varsets. In producing most common charts, we do not notice this step. When we implement log scales, however, we notice it immediately. We must log our data before averaging logs. Even if we do not compute nonlinear transformations, however, we need to specify a measurement model.

The measurement model determines how distance in a frame region relates to the ranges of the variables defining that region. Measurement models are reflected in the axes, scales, legends and other annotations that demarcate a chart's frame. Measurement models determine how values are represented (for example, as categories or magnitudes) and what the units of measurement are.

In constructing scales for statistical charts, we need to know the function used to assign values to objects. S.S. Stevens developed a taxonomy of such functions based on axioms of measurement.[41] Stevens identified four basic scale types: *nominal*, *ordinal*, *interval*, and *ratio*. These scales are widely cited in introductory statistics books and in some visualization schemes.[42]

For graphics grammar, we employ a more restrictive classification based on units of measurement. Unit scales permit standardization and conversion of metrics and raise exceptions when improper blends are attempted. The International System of Units (SI) unifies measurement under transformation rules encapsulated in a set of base classes.[43] Most of the measurements in the SI system fit within the interval and ratio levels of Stevens' system. There are other scales fitting Stevens' system that are not classified within the SI system, however. These involve units such as *category* (state, province, country, color, species, and so on), *order* (rank, index) and *measure* (probability, proportion, percent, and so on). Also, there are some additional scales that are in neither the Stevens nor the SI system, such as *partial order*.

Statistics

The statistics component receives a varset, computes statistical summaries and outputs another varset. In the simplest case, the statistical method is an identity. We do this for scatterplots. Data points are input and the same data points are output. In other cases, such as histogram binning, a varset with n rows is input and and a varset with k rows is output, where k is the number of bins ($k < n$). With smoothers (regression or interpolation), a varset with n rows is input and a varset with k rows is output, where k is the number of knots in a mesh over which smoothed values are computed. With point summaries (means, medians, and so on), a varset with n rows is input and a varset with one row is output. With regions (confidence intervals, ranges, and so on), a varset with n rows is input and a varset with two rows is output.

Statistical methods are members of their own object, so they are independent of the other elements in the system. There is no necessary connection between regression methods and curves or between confidence intervals and error bars or between histogram binning and histograms. We can represent the same statistic with a variety of different geometric objects.

Geometry

Geometric graphs are produced by graphing functions $F : B^m \to \mathbb{R}^n$ that injectively map a m-dimensional bounded region to an n-dimensional real space and have geometric names like *line*() or *tile*(). A geometric graph is the image of F. Geometric graphs are not visible; they are geometric sets.

- The *point*() graphing function produces a geometric point, which is an n-tuple. This function can also produce a finite set of points, called a *multipoint* or a *point cloud*. The set of points produced by *point*() is called a *point* graph.
- The *line*() graphing function is a bit more complicated. Let B^m be a bounded region in $\mathbb{R}^m$. Consider the function $F : B^m \to \mathbb{R}^n$, where $n = m + 1$, with the following additional properties:

  1. the image of F is bounded, and
  2. $F(x) = (\boldsymbol{v}, f(\boldsymbol{v}))$, where $f : B^m \to \mathbb{R}$ and $\boldsymbol{v} = (x_1, \ldots, x_m) \in B^m$.

  If $m = 1$, this function maps an interval to a functional curve on a bounded plane; and if $m = 2$, it maps a bounded region to a functional surface in a bounded 3D space. The *line*() graphing function produces these graphs. Like *point*(), *line*() can also produce a finite set of lines. A set of lines is called a multiline. We need this capability for representing multimodal smoothers, confidence intervals on regression lines and other multifunctional lines.
- The *path*() graphing function is similar to a line, but it is not ordered on x. A *path*() produces a *path* that connects points such that each point touches no more than two line segments. Thus, a path visits every point in a collection of points only once. If a path is closed (every point touches two line segments), we call it a circuit.
- The *area*() graphing function produces a graph containing all points within the region under the *line* graph.
- The *bar*() graphing function produces a set of closed intervals. An interval has two ends. Ordinarily, however, bars are used to denote a single value through the location of one end. The other end is anchored at a common reference point (usually zero).
- The *histobar*() graphing function produces a histogram element. This element behaves like a bar except a value maps to the area of a histobar rather than to its extent. Also, histobars are glued to each other. They have non-zero measure over their domain (even when their height is zero), so they cover an interval or region, unlike bars.
- A *schema* is a diagram that includes both general and particular features in order to represent a distribution. We have taken this usage from Tukey,[1] who invented the schematic plot, which has come to be known as the box plot because of its physical appearance. The *schema*() graphing function produces a collection of one or more points and intervals.
- The *tile*() graphing function tiles a surface or space. A *tile* graph covers and partitions the bounded region defined by a frame; there can be no gaps or overlaps between tiles. The Latinate *tessellation* (for tiling) is often used to describe the appearance of the tile graphic.
- A *contour*() graphing function produces contours, or level curves. A *contour* graph is used frequently in weather and topographic maps. Contours can be used to delineate any continuous surface.
- The *edge*() graphing function joins points with line segments (edges). Although edges join points, a point graph is not needed in a frame in order to make an edge.

Coordinates

The most popular types of charts employ Cartesian coordinates. The same real tuples in the graphs underlying these graphics can be embedded in many other coordinate systems, however. The nViZn platform on which AutoVis is based has several non-rectangular coordinate functions available, including polar, fisheye, and geographic (spherical) projections.

Aesthetics

An *aesthetic* is a function that maps a graph to a perceivable graphic. Seven of the aesthetic functions in GoG are derived from Bertin's *visual variables*: *position* (position), *size* (taille), *shape* (forme), *orientation* (orientation), *brightness* (valeur), *color* (couleur) and *granularity* (grain). In GoG, color is separated into three components. Additional GoG aesthetics involve dimensions such as blur, sound and motion.

Appendix B. Computing Scagnostics

For more detail on the material in this section, see Wilkinson *et al.* [26]

Geometric graphs

The scagnostic measures are based on the following definitions. A *graph* $G=(V,E)$ is a set V (called *vertices*) together with a relation on V induced by a set E (called *edges*). An edge $e(v,w)$, with $e \in E$ and $v,w \in V$, is a pair of vertices. A *geometric graph* $G^\star = [f(V), g(E), S]$ is an embedding of a graph in a metric space S that maps vertices to points and edges to straight line segments connecting pairs of points. We restrict our graphs to 2D Euclidean geometric graphs and omit the asterisk in subsequent notation.

The measures are derived from several features of 2D Euclidean geometric graphs. The length of an edge, *length*(e), is the Euclidean distance between its vertices. The length of a graph, *length*(G), is the sum of the lengths of its edges. A *path* is a list of successively adjacent, distinct edges. A path is *closed* if its first and last vertex are the same. A *polygon*, P, is a region bounded by a closed path. A *simple polygon* is a polygon bounded by exactly one closed path that has no intersecting edges. We restrict P to simple polygons. The perimeter of a simple polygon, *perimeter*(P), is the length of its boundary. The area of a simple polygon, *area*(P) is the area of its interior.

Minimum spanning tree

A *tree* is a graph in which any two nodes are connected by exactly one path. A *spanning tree* is an undirected tree. A *minimum spanning tree* (MST) is a spanning tree whose total length is a minimum among the lengths of all spanning trees on a given set of points.[45] We restrict ourselves to the geometric MST computed from Euclidean distances between points in a 2D Euclidean geometric graph.

Convex hull

A *hull* of a set of points embedded in 2D Euclidean space is a collection of the boundaries of one or more simple polygons that have a subset of the points for their vertices and that collectively contain all the points. This definition includes entities that range from the boundary of a single simple polygon to a collection of boundaries of simple polygons each consisting of a single point. A hull is *convex* if it contains all the straight line segments connecting any pair of points in its interior. A *peeled convex hull* is a convex hull computed after deleting points on the convex hull.

Alpha hull

An *alpha hull* is a non-convex hull based on a proximity graph. A *proximity graph* (or *neighborhood graph*) is a geometric graph whose edges are determined by an indicator function based on distances between a given set of points in a metric space.[46] To define this indicator function, we use an open disk D. We say D *touches* a point if that point is on the boundary of D. We say D *contains* a point if that point is in D. We call an open disk of fixed radius $D(r)$.

An *alpha complex* is a collection of one or more simple polygons defined by a fixed open disk.[47] For this complex, an edge exists between any pair of points that can be touched by an open disk $D(\alpha)$ containing no points. An alpha hull is the boundary (or boundaries) of an alpha complex.

Marchette[48] recommends a value of α to be the average value of the edge lengths in the MST. To reduce noise, we use a larger value, namely the 90th percentile of the MST edge lengths. We clamp this value at one-tenth the width of a frame if the percentile exceeds a tenth. This prevents us from including sparse or striated point sets in a single alpha graph.

Preprocessing

We bin our data and delete outliers before computing geometric graphs. This preprocessing improves performance of our algorithms and robustness of our measures.

Binning

We begin by normalizing the data to the unit interval and then use a 40×40 hexagonal grid to aggregate the points in each scatterplot. If there are more than 250 non-empty cells, we reduce the bin size by half and rebin. We rebin until there are no more than 250 non-empty cells. The choice of bin size is constrained by efficiency (too many bins slow down calculations of the geometric graphs) and sensitivity (too few bins obscure features in the scatterplots).

We use hexagon binning[20] to improve performance. Hexagon binning reduces anisotropy of local neighborhoods because of the near-circular shape of hexagons. This anisotropy reduction is important for keeping scagnostics orientation-independent.

Binning, like other aggregation methods, can affect statistical estimates. A well-known instance of such an effect is the ecological correlation.[49] Consequently, we apply a stabilizing transformation on some of the scagnostics computed from binned data to attenuate the influence of binning. Our weight function is

$$w = 0.7 + \frac{0.3}{1+t^2} \tag{B.1}$$

where $t = n/500$. This function is fairly constant for $n > 2000$. We determined its shape and parameters by hex binning and computing scagnostics on a wide variety of data sets. We use this function to adjust for bias in the Skewed, Sparse and Convex scagnostics formulas given below.

Deleting outliers

We delete outliers to improve robustness of our scagnostics. Classical outlier detection methods[50] are of little use for this purpose because they presume parametric densities. To avoid distributional assumptions, Tukey[51] used the recursively peeled convex hull to delete extreme points. For 1D points, this amounts to Winsorizing,[52] or successive symmetric trimming of extreme observations.

Because we do not assume convex support for our point sets, we cannot expect outliers will be outside the edges of a peeled convex hull. We want to identify points located in relatively sparse interior regions, for example. Consequently, we peel the MST instead of the convex hull. We consider an outlier to be a vertex whose adjacent edges in the MST all have a weight (length) greater than F_{inner+}, where

$$F_{inner+} = q_{75} + 1.5(q_{75} - q_{25}) \tag{B.2}$$

where q_{75} is the 75th percentile of the MST edge lengths and the expression in the parentheses is the *interquartile range* of the edge lengths.

Computing scagnostic measures

We now present the scagnostic measures computed on our three geometric graphs. In the formulas below, we use H for the convex hull, A for the alpha hull and T for the minimum spanning tree. We are interested in assessing three aspects of scattered points: *density*, *shape* and *association*.

Density measures

The following measures detect different point densities.

- *Outlying*: The Outlying scagnostic measures the proportion of the total edge length of the minimum spanning tree accounted for by the total length of edges adjacent to outlying points (as defined above). We do this calculation before deleting outliers for the other measures.

$$c_{outlying} = length(T_{outliers})/length(T) \tag{B.3}$$

- *Skewed*: We use two other density measures. The first is a relatively robust measure of skewness in the distribution of edge lengths of the MST.

$$q_{skew} = (q_{90} - q_{50})/(q_{90} - q_{10}) \tag{B.4}$$

Because Skewed tends to *decrease* with n after adaptive binning, we invert the weight in (B.1) to compute the Skewed scagnostic.

$$c_{skew} = 1 - w(1 - q_{skew}) \tag{B.5}$$

- *Sparse*: The second edge-length statistic, Sparse, measures whether points in a 2D scatterplot are confined to a lattice or a small number of locations on the plane. This can happen, for example, when tuples are produced by the product of categorical variables. It can also happen when the number of points is extremely small. We choose the 90th percentile of the distribution of edge lengths in the MST. This is the same value we use for the α statistic.

$$c_{sparse} = wq_{90} \tag{B.6}$$

where w is the weight function in (B.1). In the extremely rare event that this statistic exceeds unity (for example, when all points fall on either of the two diagonally opposing vertices of a square), we clamp the value to 1.

- *Clumpy*: An extremely skewed distribution of MST edge lengths does not necessarily indicate clustering of points. For this, we turn to another measure based on the MST: the RUNT statistic.[53] The runt size of a dendrogram node is the smaller of the number of leaves of each of the two subtrees joined at that node. Since there is an isomorphism between a single-linkage dendrogram and the MST,[54] we can associate a runt size (r_j) with each edge (e_j) in the MST, as described by Stuetzle.[55] The RUNT graph (R_j) corresponding to each edge is the smaller of the two subsets of edges that are still connected to each of the two vertices in e_j after deleting edges in the MST with lengths less than $length(e_j)$.
The RUNT-based measure responds to clusters with small maximum intra-cluster distance relative to the length of their nearest-neighbor inter-cluster distance. In the formula below, j runs over all edges in T and k runs over all edges in R_j.

$$c_{clumpy} = \max_j \left[1 - \max_k[length(e_k)]/length(e_j)\right] \tag{B.7}$$

- *Striated*: We define coherence in a set of points as the presence of relatively smooth paths in the minimum spanning tree. Smooth algebraic functions, time series and curves (for example, spirals) fit this definition. So do points arranged in flows or vector fields. Another common example is the pattern of parallel lines of points produced by the product of categorical and continuous variables.
We use a measure based on the number of adjacent edges in the MST whose cosine is less than −0.75. Let $V^{(2)} \subseteq V$ be the set of all vertices of degree 2 in V and let $I()$ be an indicator function. Then

$$c_{striate} = \frac{1}{|V|} \sum_{v \in V^{(2)}} I(\cos\theta_{e(v,a)e(v,b)} < -0.75) \tag{B.8}$$

Shape measures

The shape of a set of scattered points is our next consideration. We want to detect if a set of scattered points on the plane appears to be connected, convex and so forth. Of course, scattered points are by definition *not* these things, so we need additional machinery (based on geometric graphs) to allow us to make such inferences. In particular, we will measure aspects of the convex hull, the alpha hull and the minimum spanning tree.

- *Convex*: Our convexity measure is based on the ratio of the area of the alpha hull and the area of the convex hull. This ratio will be 1 if the non-convex hull and the convex hull have identical areas.

$$c_{convex} = w[area(A)/area(H)] \tag{B.9}$$

where w is the weight function in (B.1).
- *Skinny*: The ratio of perimeter to the area of a polygon measures, roughly, how skinny it is. We use a corrected and normalized ratio so that a circle yields a value of 0, a square yields 0.12 and a skinny polygon yields a value near 1.

$$c_{skinny} = 1 - \sqrt{4\pi area(A)}/perimeter(A) \tag{B.10}$$

- *Stringy*: A stringy shape is a skinny shape with no branches. We count vertices of degree 2 in the minimum spanning tree and compare them to the overall number of vertices minus the number of single-degree vertices.

$$c_{stringy} = \frac{|V^{(2)}|}{|V| - |V^{(1)}|} \tag{B.11}$$

We cube the Stringy measure to adjust for negative skew in its conditional distribution on n.

Association measure

We are interested in a symmetric and relatively robust measure of association.

- *Monotonic*: We use the squared Spearman correlation coefficient to assess monotonicity in a scatterplot. We square the coefficient to accentuate the large values and to remove the distinction between negative and positive coefficients. We assume investigators are most interested in strong relationships, whether negative or positive.

$$c_{monotonic} = r^2_{spearman} \tag{B.12}$$

This is the only coefficient not based on a subset of the Delaunay graph.