Radial Visualizations for Comparative Data Analysis

Geoffrey M. Draper and Richard F. Riesenfeld

School of Computing, University of Utah, Salt Lake City, Utah, USA

Abstract
We present a visual language for performing queries on large multivariate data sets. The workspace consists of an interactive canvas upon which attributes and values can be added and removed via direct manipulation. Both the specification of queries and the visualization of results occur in real time on the same canvas. We also describe our prototype software system, SQiRL, which implements this metaphor in a three-tiered fashion. First, the default view provides a simple-to-learn interface for query evaluation. Second, intermediate users are provided a straightforward method for comparing the results of two queries. Third, advanced users can make use of a “radial crosstab,” a new interactive visualization technique that melds the expressive power of crosstabulation with a drag-and-drop canvas. We demonstrate our system’s generality by showing its application to a variety of data sets.

Key words: Visual query languages, radial visualization, crosstabulation

1. Introduction

One goal of the “information visualization for the masses” [40] movement is to make easy-to-use tools available for people who are not necessarily experts in data analysis. At the same time, system builders must be careful not to dilute the interface to the point that it inhibits expert users from performing their work. A compromise, as proposed by Norman [31], is to design systems such that advanced features are at first hidden. As users’ experience and confidence grow, additional functionality is revealed.

For command-line interfaces (CLI), this is trivial. When users first start with the interface, they use only the bare minimum set of commands required for their task. As the users’ proficiency increases, new commands are learned as the need arises. In a WYSIWYG interface, the situation is more complicated because traditional menu systems and toolbars present all available
options to the user at once. Finding the right command in a sea of hierarchical menus, wizards, and “helpful” dialogs can be frustrating for new users. The most common solution is to hide advanced options in a collapsable tree widget, or to omit rarely-used options from a pull-down menu.

While this approach has been successfully used in traditional WIMP-style GUI’s, it is less applicable for the growing class of canvas-based interfaces commonly found in information visualization systems. While a canvas-based interface will often have menus and toolbars for incidental needs, the core functionality is usually accessed through the direct manipulation of on-screen objects, rather than by picking a command from a list. Neither the CLI’s “hide all” mentality nor the GUI’s “show all” approach is best here. As such, a middle-ground approach is called for. We address this issue by showing the basic interface by default, and allowing the user to “upgrade” to a more advanced canvas upon request. We employ a multi-tabbed layout for these separate canvases, similar to those found in modern web browsers. The default (simple) visualization is always available, and the user can switch to and from the advanced canvases as desired.

This paper describes an application of this concept to a visual query system for demographic data analysis. In [10], we described an early version of our prototype implementation, SQiRL (Simple Query Interface with a Radial Layout). SQiRL provides an easy-to-learn user interface for exploring relationships in multivariate data. However, in order to maintain a simple design, we purposely limited its generality in the first iteration of the software. In particular, it lacked the ability to simultaneously compare intersections among multiple variables.

In this paper, we show how SQiRL’s unique visual query metaphor can be extended to support more advanced kinds of analysis, without sacrificing the simplicity that makes SQiRL’s basic interface appealing for novices. We propose a three-tiered visualization interface tailored to the needs of beginning, intermediate, and advanced users. In the basic visualization, queries can be constructed quickly and easily via a drag-and-drop interface. For intermediate users, we provide a straightforward “visual diff” [38] visualization for comparing the results of two related queries. Advanced users have the ability to evaluate the results of several multivariate queries simultaneously. In each case, the visualizations are opened in separate tabs, so that each canvas shows only the view particular to it. This has at least two advantages. First, by making the user explicitly request a more advanced mode, we keep the default UI simple for beginners. Second, if a beginning user
does inadvertently access an undesired mode, the original interface can be restored simply by re-selecting the original tab.

The remainder of this paper is organized as follows. In Section 2, we review some related research in visual database query languages, as well as the growing trend towards radial visualization techniques. Section 3 contains a brief review of the design and usage of SQiRL for performing basic queries. In Section 4, we extend this basic usage to include the comparison of two related queries. Section 5 discusses a simple yet powerful extension to SQiRL for advanced data analysis. As the original motivating reason for developing the SQiRL system was for the analysis of political opinion polls, we use the NES 2004 data set [29] in most of the examples shown in this paper. However, Section 8 demonstrates this system’s generality by showing its application to two other data sets. Finally, in Section 9, we present our conclusions and discuss some possible avenues for further work.

2. Related Work

This work draws from two areas of active research in computer science, in particular, visual query languages for databases and radial visualization. We now present a brief review of each.

2.1. Visual Query Languages

Making databases easier to use has been a subject of research for several decades. One of the first such efforts that is still in widespread use is Query By Example [52]. Cammarano et al. [3] make the observation that most user interfaces for databases take either one of two approaches. Either the user interface aids in formulating the query, or in visualizing the results.

Catarci et al. [4] provide a comprehensive, if somewhat dated, survey of the literature in the first category. Of particular interest is the problem of querying semistructured or unstructured data, such as websites or emails. Sinha and Karger [41] propose a system for aiding in navigation of semistructured data sets by suggesting navigation hints to the user. Trigoni [47] lets the user refine a query over time by gradually disclosing the underlying data one part at a time. Goldman and Widom [16] propose a method for exploiting similarities among pages in the same website to perform more effective queries. Polyviou et al. [35] describe an interface for performing database queries based on the ubiquitous filesystem browser interface. The VisTrails
system [38] makes use of provenance data to maintain a history of past queries for creating visualizations.

In the second category, there are many systems which offer a direct-manipulation interface for browsing the results of a query. Furnas and Rauch [15] as well as Stonebraker [44] present canvas-based visualizations that support zooming and panning. XmdvTool [48] facilitates the creation of standard statistical graphs to display query results. Visage [36] by Roth et al. is a highly interactive direct manipulation system that uses a variety of graphing techniques to communicate results to the user. ManyEyes [49] and Swivel [45] are websites for collaborative visualization. In addition, Keim [23] and de Oliveira and Levkowitz [7] present surveys of many database visualization techniques. While not necessarily interactive, parallel coordinates [20] and scatterplots [50] also support the simultaneous visualization of many variables. Elmqvist et al. [11] offer an interactive scatterplot that uses animated transitions between queries.

In contrast, Tableau [46] is a commercial system incorporating both a novel query interface and a variety of integrated visualizations. Similar to a pivot table in a spreadsheet, Tableau allows users to view correlations in the data with respect to any particular attribute in the data set.

Our work differs from existing systems primarily in our strict focus on ease of use and interactivity. Some systems go to great lengths to match the expressive power of SQL, but in so doing, effectively put their work out of the reach of users who have neither the time nor the inclination to learn to use a full-fledged DBMS, graphical or otherwise. In contrast, our approach supports a narrower range of queries, focusing on those commonly used to analyze trends in demographic data. To our knowledge, SQiRL is the first visual query language which uses the radial layout metaphor for both querying and visualization. We review related work in radial user interfaces in the next section.

2.2. Radial user interfaces

An increasingly popular metaphor in contemporary information visualization, radial charting techniques nonetheless have a tradition spanning back hundreds of years. Pre-digital examples of radial information layouts include William Playfair’s 1801 invention of the pie chart [34, 42], and Florence Nightingale’s rose diagrams [30] for communicating sanitary conditions in British military hospitals during the Crimean War. In the mid-twentieth
century, Northway used radial diagrams to track the social behaviors of grade-school children [32].

Much of the recent work in radial user interfaces traces its lineage to research in graph layout algorithms for computer graphics [18]. These algorithms, in turn, have inspired techniques for visualizing multivariate data. Many such designs involve positioning data points as nodes on the spokes on a wheel [17, 19]. In these visualizations, the center point of the canvas holds some semantic meaning, and the distance of each node from the center shows a relationship relative to it. Recent examples of this are DataRose [12] and Vectorized Radviz [39].

In contrast, a second variety of radial visualization (called radial space filling or RSF [43]), the data points are typically arranged in compact concentric rings [25], and rendered so as to form a circle. Each ring represents a different attribute of the data. Examples include polar treemaps [22], fan charts [9], and Radial Traffic Analyzer [24].

Another general category of radial visualization arranges the data points around the circumference of a ring, while reserving the interior of the ring for other data. Relationships between data points are often rendered as lines between nodes on the circumference and nodes in the interior. Examples include Daisy [6], VisAlert [27], and Manager’s Dartboard [8].

Our method is most closely related to the latter category. Our approach differs from previous work in that we reserve the interior of the ring for constructing user-specified queries rather than for rendering line segments between related entities. Instead, correlations in the data are displayed in a series of curved bar charts on the ring’s circumference.

3. Basic Queries

The basic UI of SQiRL provides an easy-to-use canvas for performing database queries and visualizing the results. We provide only a brief review of its interaction metaphor here. For full details, please see our earlier paper [10].

The canvas itself (Figure 1) is a workspace upon which icons representing attributes and values can be manipulated. The most prominent feature of the canvas is the large ring in the center. We divide the canvas into three distinct regions: the ring itself, the area inside the ring, and the area outside the ring.
• **Ring.** The ring is divided up into multiple equiangular sectors, each representing one of the attributes in the data set. Each sector is partitioned into two concentric layers. The outer layer of the sector displays the name of the attribute. The inner layer contains a sequence of subsectors that create a (curved) stacked bar chart [50]. Each subsector is scaled in direct proportion to the number of entities exhibiting that attribute value, and displays the corresponding name and percentage. By default, the percentages are computed relative to the entire population, unless the population is restricted by placing values into the interior of the ring.

• **Interior.** By dragging specific values for an attribute into the ring’s interior, the user can selectively refine the population on which the displayed percentages are based. For example, if the value Democrat from the attribute Party Affiliation is placed inside the ring, then the results displayed on the ring will reflect only those entities whose party affiliation is listed as Democratic.

• **Exterior.** The area outside the ring simply serves as a working storage area, or cache, for attributes and values that are not currently part of any query, but which recently were or soon may be. Attributes and
values that are not likely to be needed soon can optionally be dragged to the side panel to clear up space on the canvas.

The ring itself shares some similarities to a multi-series donut chart [13]. With this visualization, however, the interior of the ring is more than a decoration; it serves as a query workspace. For each value inside the ring, the sample population is refined by ANDing the values together. For example, if the values Democrat, Female, and Married (from the attributes Party Affiliation, Gender, and Marital Status, respectively) were placed in the ring’s interior, then the percentages displayed on the ring would reflect the subset of the total population that fits the description of married women who are Democrats (Fig. 2). If two or more values from the same attribute are placed inside the ring (for example, Self-employed and Retired from the attribute Employment Status), they would be ORed together instead of ANDed.

Each time a value is dragged into or out of the ring’s interior, the sectors on the ring are updated to reflect the values for the population specified in the current query. The transition from the previous to the current query’s
results is smoothly animated to help the user maintain a sense of context [51]. The queries take place in real time, enabling interactive data exploration and rapid testing of hypotheses.

Likewise, any attribute in the data set can be dragged on or off the ring itself, letting the user examine the data from the vantage point of any variable. In some ways, this behavior is reminiscent of a pivot table in a spreadsheet [13], albeit with an arguably simpler interface.

In effect, SQiRL is a direct manipulation interface for specifying two complementary kinds of queries.

1. By dragging an attribute value into (or out of) the ring’s interior, the user restricts the query to a subset of the total population that matches certain characteristics.

2. By dragging an attribute onto (or off of) the ring’s circumference, the user specifies the attributes for which quantitative information is desired, regarding the given population.

Stated another way, the values inside the ring could be considered independent variables in that they determine the results of the dependent variables on the ring’s circumference.

4. Visual Diffs

After users become comfortable with the basic interface, we have observed that the first thing most users want is to compare the results of two queries. For example, if one query shows what percentage of men in a given population voted for a certain candidate, a logical follow-up query might be to see what percentage of women in the same population voted the same way. The original SQiRL prototype afforded no way to do this, other than to perform the two queries in sequence and hope to remember the results from one query to the next. Clearly, this approach leaves ample room for improvement.

We considered a number of interfaces for comparing multiple queries, such as: rendering visualizations side by side, overlapping visualizations, concentric rings, and time-delayed animations. We based our evaluation of these techniques on three main criteria: **Flexibility**: Does the visualization allow the comparison of arbitrarily constructed queries, or does it impose a similarity constraint on the two queries? **Ease of Comparison**: Does the visualization make it easy to spot differences between the two queries? **Compactness**: Does the visualization make efficient use of limited real estate on the display?
Figure 3: Proposed “side by side” visual diff. Although the charts shown above use the same set of attributes on the ring’s circumference, this is not a strict requirement for side by side comparisons. This affords the side by side approach great flexibility.

We present our findings below.

- **Side by side.** Perhaps the most obvious way of comparing two results is simply to render two charts and display them side by side (Figure 3). The side-by-side comparison technique is at the heart of the spreadsheet interface for visualization exploration as proposed by Jankun-Kelly and Ma [21]. This technique is very flexible, as it places no restrictions on the content of the queries being compared. The queries may differ by only one or two terms, or they may be highly dissimilar; the side-by-side approach makes it possible to compare any two queries regardless of similarity. Another consequence of this approach is that its “real estate” needs grow linearly with the number of comparisons being made. The space requirements can be relieved somewhat by decreasing the resolution at which the chart is rendered; however, this smaller size may make it more difficult for the user to interpret the results.

- **Partial overlap.** A slight modification to the side-by-side approach is to render two charts, but position them so that they overlap. This decreases the amount of space required to render the visualization, at the cost of some occlusion. The occlusion problem can be ameliorated somewhat, however, by rendering one or both of the charts with partial transparency (Figure 4). Like the side-by-side comparisons, the partial overlap technique allows a high degree of freedom with regard to the contents of the queries being visualized.
Figure 4: Proposed “partial overlap” visual diff. While less demanding in terms of space, the occlusion issues inherent to this approach make it unsuitable for general use.

Figure 5: Proposed “concentric” visual diff. This is more space-efficient than the previous two techniques, and simplifies the problem of directly comparing sector magnitude. However, the manner of showing which population constraints apply to which ring is somewhat awkward.
Table 1: Comparison of radial “visual diff” strategies

<table>
<thead>
<tr>
<th></th>
<th>Flexibility</th>
<th>Ease of use</th>
<th>Compactness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Side by side</td>
<td>Good</td>
<td>Poor</td>
<td>Poor</td>
</tr>
<tr>
<td>Partial overlap</td>
<td>Good</td>
<td>Poor</td>
<td>Fair</td>
</tr>
<tr>
<td>Concentric rings</td>
<td>Poor</td>
<td>Fair</td>
<td>Fair</td>
</tr>
<tr>
<td>Animated</td>
<td>Poor</td>
<td>Good</td>
<td>Good</td>
</tr>
</tbody>
</table>

- **Concentric rings.** Another way of showing the results of multiple queries is to render two rings concentrically, one ring per query result. In this way, each attribute/value pair for the various subpopulations is rendered adjacently, facilitating comparison. While this approach uses less real estate than the previous two techniques, it is inflexible in that its effectiveness depends on each query using the same set of attributes on ring’s circumference. Moreover, it is not obvious how one should display the values inside the ring, so that each query’s terms are distinct. Our current approach, which renders the filter terms as small Euler diagrams, is admittedly somewhat clumsy (Figure 5).

- **Animated.** In the animated comparison scheme, the display toggles between two query results. The results of one query are shown for a moment, then the sectors smoothly resize to show the results of the other query. Like the concentric approach, the animated method is most effective when the set of attributes along the ring’s circumference is the same across both queries. However, in the case of animated difference visualization, the terms in the ring’s interior are easily distinguished from one query to the other by a “fade-in/fade-out” effect during the animation (Figure 6). Of all the comparison techniques discussed herein, this one makes the most efficient use of display real estate, since it uses time, rather than space, to communicate difference information. This visualization also compares favorably with the others with respect to usability — users typically need no explanation of how to interpret the animated chart.

These four visualization techniques and their relative strengths are summarized in Table 1. Although none of them is perfect, the animated approach
seems to strike the best compromise among the competing virtues of flexibility, ease of use, and compactness. Accordingly, we have chosen the animated method as the primary difference visualization metaphor in the SQiRL system.

5. Radial Crosstabulation

Our third visualization is more advanced, and is intended for experienced users. It is based on the metaphor of crosstabulation, a common method of statistical analysis [33]. In a crosstabulation (or crosstab), two variables are displayed on the axes of a table. Each row and column represents a different possible value for that variable. Each cell displays the number of times that the combination of the values shown in the corresponding row/column occurs.

Each cell in a crosstab typically contains a count, a percentage, or both. Table 2 is an example of a simple crosstab. It shows the relationship between political ideology in U.S. voters and how they voted in the 2004 U.S. presidential election. If row percentages are used, then the percentages in each row add up to 100%. Similarly, for column percentages, the totals for each cell reflect a percentage of the column’s total. Table 2 utilizes row percentages, indicating the percentage of votes that each candidate received, per ideological group.

A crosstab in its basic form displays relationships between two and only two variables. Additional relationships can also be shown via the addition of control variables, at the expense of a somewhat more complicated table. Table 3 is an enhanced version of Table 2, with the addition of a control
Table 2: Political Ideology versus Vote for President (2004)

<table>
<thead>
<tr>
<th></th>
<th>Kerry</th>
<th>Bush</th>
<th>Nader</th>
<th>Other</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Liberal</td>
<td>155 (89.6%)</td>
<td>15 (8.7%)</td>
<td>1 (0.6%)</td>
<td>2 (1.2%)</td>
<td>173 (100%)</td>
</tr>
<tr>
<td>Moderate</td>
<td>106 (55.2%)</td>
<td>84 (43.8%)</td>
<td>1 (0.5%)</td>
<td>1 (0.5%)</td>
<td>192 (100%)</td>
</tr>
<tr>
<td>Conservative</td>
<td>50 (16.3%)</td>
<td>250 (81.7%)</td>
<td>2 (0.7%)</td>
<td>4 (1.3%)</td>
<td>306 (100%)</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>311 (46.3%)</td>
<td>349 (52.0%)</td>
<td>4 (0.6%)</td>
<td>7 (1.0%)</td>
<td>671 (100%)</td>
</tr>
</tbody>
</table>

Table 3: Political Ideology (with Children in Household) versus Vote for President (2004)

<table>
<thead>
<tr>
<th></th>
<th>No Children in Household</th>
<th>≥1 Children in Household</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Liberal</td>
<td>127 (89.4%)</td>
<td>15 (8.5%)</td>
<td>1 (0.7%)</td>
</tr>
<tr>
<td>Moderate</td>
<td>28 (90.3%)</td>
<td>3 (9.7%)</td>
<td>0 (0%)</td>
</tr>
<tr>
<td>Conservative</td>
<td>39 (15.9%)</td>
<td>201 (81.7%)</td>
<td>2 (0.8%)</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>75 (53.2%)</td>
<td>64 (45.4%)</td>
<td>1 (0.7%)</td>
</tr>
</tbody>
</table>

variable (the presence or absence of children in the household).

The radial query language described in Sections 3 and 4 can be extended to create crosstabulations in a space-efficient way. An example is shown in Figure 7. The row labels are shown in the top-center of the ring, and the “columns” are projected radially around the circumference. For improved readability, the column labels are printed in each sector, rather than only in the column header.

To illustrate the benefit of a radial crosstab, let’s say that we want to evaluate the relationship between one’s affiliation with a certain political party and a number of other variables: annual income, education level, and how one voted in the 2004 general election. Using traditional methods, this would require the creation of three separate crosstabs. The same information can be shown in just one radial crosstab diagram (Figure 8). While it is true that this approach allocates less space per variable, our use of stacked bar charts ensures that the most prominent values for each variable will still
be visible. As always, “mousing over” any portion of the chart reveals full details about a given value or variable.

As described in Section 3, the query can be limited to specific subpopulations by placing icons representing values within the ring’s interior. This action is likewise available in SQiRL’s radial crosstab view. This is analogous to adding control variables to a crosstab. However, the radial approach is arguably more flexible because it allows any number of values to be placed in the ring’s interior (Figure 9), whereas a traditional crosstab only (easily) accommodates a single control variable. In many ways, the radial crosstab is a generalization of the “concentric” visual diff described in Section 4, with the additional benefit of interactivity.

Perhaps the greatest strength of SQiRL is its simplicity. Creating complex tables like those shown in Tables 2 and 3 with current state-of-the-art statistics software is a multi-step procedure, requiring the navigation of a series of menu options and dialogs. Once a chart is created, it is static and non-interactive. In contrast, our method supports the creation of radial charts with similar expressive power, via a straightforward “one step” drag and drop interface. Furthermore, our radial crosstabs are fully interactive. Variables can be quickly and easily added and removed from both the circum-

---

Figure 7: Very simple radial crosstab. The “rows” are shown in the wedge at the top of the ring, while the “columns” are projected radially around the circumference.
Figure 8: More advanced radial crosstab. Multiple variables are projected around the circumference, creating a series of stacked bar charts in every ring. Due to space limitations, labels for smaller sectors are truncated, allowing sectors of greater radial magnitude to receive due emphasis.

Figure 9: By introducing additional constraints (control variables) into the ring’s interior, we can further specify the subpopulation being analyzed. The sectors in each ring update to their correct sizes via a smooth animation to help the user maintain context.
ference and the interior, and the chart updates in real time. In this way, our visualization scheme encourages a more fluid workflow in which hypotheses can be rapidly tested and iteratively revised.

6. Evaluation

The visualizations available in the SQiRL system are intrinsically multi-tiered for beginning, intermediate, and advanced users. The basic query method is suitable for beginning users; indeed, prior work [10] has demonstrated that people with little or no experience in data analysis can quickly learn how to perform queries in SQiRL. Once users are comfortable with performing basic queries, we have observed that they typically ask for a way to directly compare the results of two queries. Our “visual diff” technique facilitates this without overcomplicating the user interface. On the other hand, advanced users who perform data analysis as part of their everyday work tend to favor a crosstab-like interface. The radial crosstabs described in Section 5 provide the expressive power of traditional crosstabs with the flexibility of SQiRL’s basic query metaphor. Each visualization canvas opens in a new tab, clearly separating the novice, intermediate, and advanced modes of operation.

As an informal evaluation of the comparative analysis techniques described in this paper, we have demonstrated the software to three political science professors, experts in demographic data analysis. The response has been quite positive. Since these scientists were already familiar with the concept of crosstabulation, it did not take them long to adjust to the radial form. They especially appreciated the radial crosstab’s interactivity, in particular, its support for iterative adjustment of control variables. In addition, each professor expressed interest in using SQiRL in the classroom, as a gentle way of introducing students to the analysis of opinion polls.

7. Implementation Notes

The SQiRL software is implemented in Java 1.6, with an embedded Derby [2] database backend. Since the UI does not depend on any particular database or schema, we have been able to reuse the visualization with a variety of data sets without modifying or recompiling the code. We discuss a few examples in the next section.
8. Application

Although originally designed for the purpose of exploring opinion poll data, we have found that SQiRL’s visual query metaphor can be applied to a variety of demographic data sets, or in fact, any categorical tabular data set. We discuss two non-polling examples below.

8.1. Student Demographics

For executives or administrators to function correctly, they must know those whom they serve. This is especially true in a university setting. Accurate demographic information guides the administration in developing programs to serve the students. Publicly-available institutional statistics, such as the Common Data Set [5] for universities, allows prospective students to make informed decisions about which school to attend. However, universities typically publish their Common Data Set only in aggregate (rather than raw) form. Thus, finding relationships among disparate variables in the data is difficult, if not impossible. When raw data is available, an interactive tool such as SQiRL could help educators (who may not be experts in data analysis), to quickly find meaningful trends in the data.

As an example, we use a partial data set of 355 current and past graduate students from a research department at a university. A simple chart such as the one in Figure 10 shows that while these students originate from many nations, the vast majority come from one of three countries: the United States, China, and India. It also shows that most students begin their education in the Fall, the traditional beginning of the scholastic year. We also see that there is a slight majority of PhD students over Masters students.

While this is interesting, it does not yet offer any value beyond what one could read in the university’s publicly-available Common Data Set. The real advantage comes from combining one or more variables to obtain insights about the student body. For example, by rearranging the chart as shown in Figure 11, we see that among female students, there is a higher percentage of internationalization than among the general student body. Among Indian students, there is a high proportion of master’s students (Figure 12a); for Chinese, the pendulum swings more to doctoral side (Figure 12b). This relationship can be more fully explored in a radial crosstab (Figure 13).

University recruiters could use this information to seek out groups of students that have been historically underserved at their institution. Similarly, faculty could use this information to tailor their programs to match
Figure 10: A snapshot of the countries of origin, semester of admittance, and degree program of 355 graduate student researchers.

Figure 11: By introducing the qualifier “Female” into the ring’s interior, the chart changes to show statistics for female students only.
Figure 12: For the students in this particular data set, Indian students are more likely to seek master’s degrees than Chinese students.

Figure 13: A radial crosstab showing the relationship between student nationality and degree program, gender, and semester of admittance.
the interests of their students (or vice versa!).

8.2. Marathon Times

Our next example looks at a data set that, while human-based, is not necessarily demographic. It shows the age, sex, and completion time for 28,764 runners in the 2001 Chicago marathon. (This data set was obtained from [33].) While the limited number of variables does not permit the same depth of analysis as in the previous data sets, interesting relationships can nonetheless be discovered.

Figure 14 provides an overview of the entire data set. By focusing on those athletes who completed the race in less than three hours, we see that most were males between the ages of 25 and 39 (Figure 15). If we instead look at those runners under the age of 25, we see that a slight majority are women, and that most finished the race in under five hours (Figure 16). From a crosstab view (Figure 17), it is clear that the 25–39 age group was dominant in every time bracket, while the proportion of women to men increased with time.
Figure 15: By confining the query to only those runners who completed the race in less than three hours, we see a breakdown for age and gender for this group.

Figure 16: Among runners under the age of 25, slightly over half are women, and a clear majority finished the race in less than five hours.
9. Discussion and Further Work

In this paper, we have presented three complementary visualization techniques as implemented in SQiRL, a prototype software system for visual data analysis. We began by reviewing a direct manipulation metaphor for querying and visualizing demographic data. We then introduced two extensions to the basic visualization method. First, we discussed several alternate schemes for comparing the results of two queries. Second, we introduced a more general visualization for comparing many variables simultaneously, based on the familiar metaphor of crosstabulation. Each of these three visualizations builds upon the other, allowing the user to progress to increasingly more advanced visualizations as he or she becomes more familiar with the tool. Moreover, our use of separate tabs for these visualizations allowed us to keep advanced features separate from the basic interface. This preserves the simplicity of the visualization for beginners (SQiRL’s primary intended audience), while allowing advanced users access to the tools they need.

While we believe radial visualization to be a compelling and valuable methodology, it remains to be seen whether the interactive techniques discussed in this paper would not apply equally well to more traditional visual-
ization schemes such as linear bar charts or line graphs.

There are several avenues for future work in this area. One valid criticism of radial visualization techniques in general is that the nodes near the periphery of the display are rendered very small [26]. This could be overcome in SQiRL by use of a fish eye lens [14, 37].

Due to our charts’ circular shape, the outer rings necessarily have a greater circumference than the inner rings. This geometric side effect has potential consequences on the way that a radial crosstab is read and interpreted. In Figure 13, for example, the sector labeled “PhD” on the Other ring has a smaller arc length than the corresponding sector on the USA ring, even though the first sector represents a higher percentage. This difference is due to the fact that arc length is derived from the radius as well as the angular magnitude. As such, those sectors on the outer rings may receive undue emphasis due to their greater radius. This issue is not limited to SQiRL alone, and affects any radial visualization system that uses concentric rings, for example [1, 28, 24].

We anticipate making SQiRL publicly available in early 2009, under a free software license. Check the author’s website for more information.

10. Acknowledgements

The authors wish to thank Pat Hanrahan for his useful comments about this work. We also thank Adam Luedtke and Baodong Liu of the University of Utah Political Science department for their feedback. Data sets were provided courtesy of American National Election Studies and the Chicago Marathon. This work was supported in part by the Scientific Computing and Imaging Institute at the University of Utah. Portions of Sections 2 and 3 originally appeared, in modified form, in [10].

References


25


[34] W. Playfair. *The statistical breviary; shewing, on a principle entirely new, the resources of every state and kingdom in Europe; illustrated with stained copper plate charts, representing the physical powers of each distinct nation with ease and perspicuity. To which is added, a similar*


