

Developing and assessing light-weight data-driven exploratory geovisualization tools for the web

Erik B. Steiner, Alan M. MacEachren, Diansheng Guo

GeoVISTA Center, Department of Geography, 302 Walker, Penn State University
maceachren@psu.edu; www.geovista.psu.edu

1 INTRODUCTION

Dramatic advances in 'mapping' have occurred during the past decade. Among the most fundamental are the change from maps that communicate a single message, statically, to maps that support exploration of multiple perspectives, dynamically. Although dynamic maps have been appearing on the WWW (world wide web – hereafter, simply web) for several years, most of the innovation in geovisualization has been directed to expert users working on desktop (or more powerful) computers. Technological and societal changes are making it possible to extend from this base toward geovisualization tools designed to support non-expert needs that can be met through highly interactive geoinformation exploration as well as the web-delivery of such tools.

In this paper, we describe and demonstrate a set of light-weight, data-driven geovisualization tools for the web that are being developed as part of a 'Digital Government' (DG) research initiative in the U.S. The focus of our contributions to the wider DG effort is on dynamic maps and graphics to support visual delivery and analysis of federal statistical summaries (compilations of numerical data, such as those from the national Census, that are published in aggregate form).

The initial section below provides some background for both the kinds of exploratory geovisualization tools being developed and for the DG project they are a part of. Then, we detail (briefly) the implementation of a web-based prototype using Macromedia Flash as the development platform. The prototype includes dynamic and linked exploratory geospatial data analysis methods (methods that are now relatively standard for systems targeted at expert analysts but, in the implementation presented here, are designed for use by non-expert users). It supports data access from a commercial database (Oracle) through the Flash (map-based) interface. We close with a brief discussion of plans for extensions to these tools and of the iterative, human-centered design approach we are taking to tool development and assessment.

2 Background

Interactive maps for the web have become quite commonplace, and substantial investments in both geospatial research and applications are being directed toward serving of geospatial data via the web (see (MacEachren, 1998) for a recent review). There have also been several successful web-implementations of exploratory

geovisualization methods (see Dykes, 1997; Andrienko and Andrienko, 1999; Harrower et al., 2000). Still, it remains a challenge to design web-based interactive geovisualization tools that meet cartographic design standards while being flexible enough to achieve the core geovisualization goal of supporting a multi-perspective approach to data exploration. This challenge generally requires that developers (and often users) master formal computer programming languages; and even the best of current tools impose serious limitations from the perspective of graphic design – limitations that are likely to impede the transitioning of multi-perspective geovisualization methods from expert to non-expert.

The goal of our particular DG project is to develop 'quality graphics' that support the understanding and flexible analysis of statistical summaries produced by eight federal government agencies in the U.S. The research has a dual focus on provision of relatively sophisticated exploratory data analysis tools to support expert work in generating these statistical summaries (e.g., to support population projections by the Census Bureau) and to support access to this information by the public (with the target audience here ranging from the ordinary citizen interested in health-environment issues to the university researcher conducting a demographic or economic analysis). The Flash-enabled tools described below are intended to support this dual focus. For the first focus, Flash is being used to provide both an environment for formal cognitive experimentation and for rapid prototyping. The rapid prototyping allows dynamic data exploration methods to be quickly implemented and assessed, before committing resources to final implementation as part of a suite of Java-based tools being constructed within *GeoVISTA Studio* (MacEachren et al., 2001) and/or Illumitek's *nViZn*. For the second focus, we are experimenting with Flash as a final web-delivery mechanism that supports user-friendly but flexible exploratory geovisualization tools that can be used effectively by non-experts.

3 Prototype

This section introduces the Flash-based web geovisualization tools we are beginning to implement and assess. It begins with discussion of design and implementation of interactive exploratory maps that support dynamic linking between visual representation forms as well as user construction of temporal and non-temporal sequences. Then, the process of linking these interactive maps to a database is detailed.

3.1 Design of dynamic maps

We have created several dynamic maps using Flash in an effort to determine capabilities and limitations of this technology for exploratory geovisualization on the web and for development and testing of rapid prototypes that might be rebuilt with extended functionality using Java. Here we illustrate, and describe briefly, three of these prototypes.

3.1.1 Linked geographic brushing

In our initial prototype, [we implemented a choropleth map and scatterplot display using state-level data of the United States](#). The interface was designed in an earlier version of Flash (4.0) that supported data access only through loading URL Encoded text files. Later prototypes (discussed in sections 3.1.2 and 3.1.3) implement more sophisticated database support. Once the data are loaded in our initial interface, they are linked to their geographic entities (states) and a scatterplot representation. This functionality is illustrated in figure 1 (below) by building a custom application using Flash's Action Script (a scripting language that is similar, conceptually, to JavaScript while emphasizing graphic applications). Flash's illustration tools allow you to design visual elements in an interface, while ActionScript allows you to control the global characteristics (color, size, location, transparency, etc.) of these elements. In the case of the choropleth map, state elements were given a lightness value that corresponded to their data value, and in the case of the scatterplot, individual points were given positions according to their values on two separate attributes.

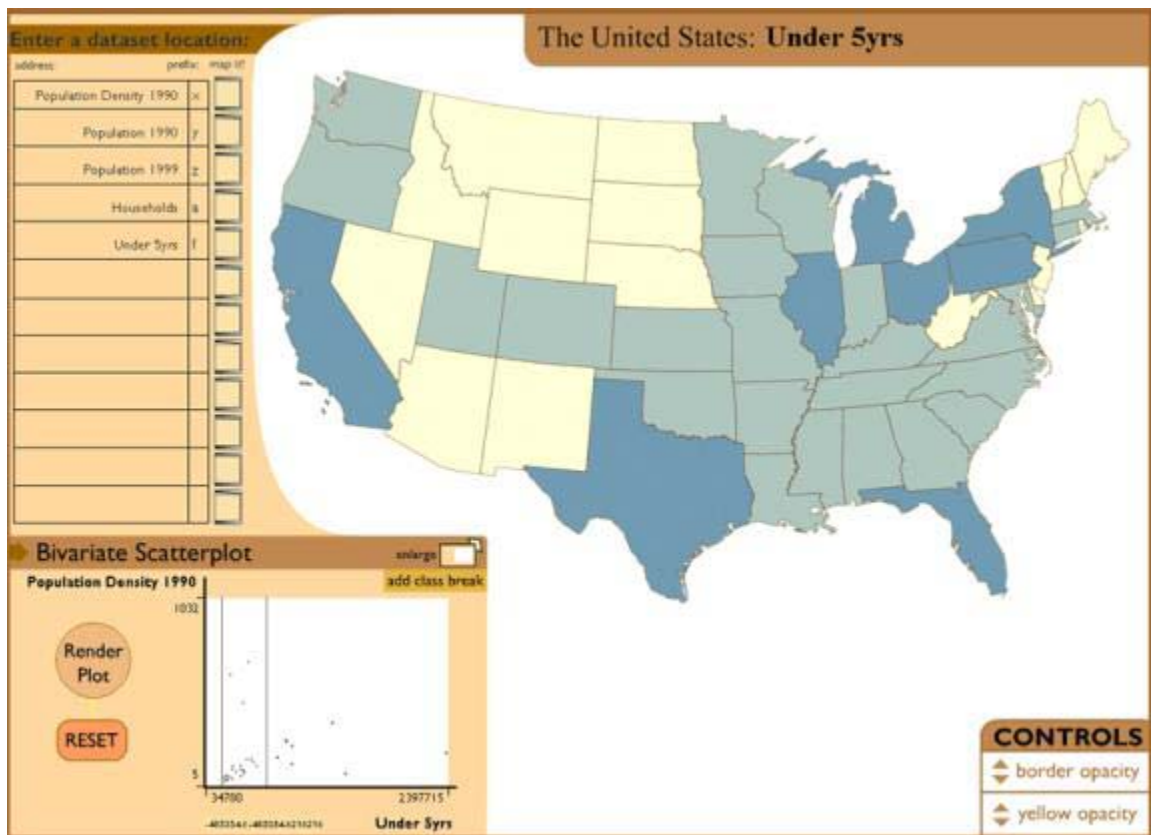


Figure 1. Prototype 1: Linked geographic brushing interface

The prototype demonstrates the concept of linking the scatterplot and choropleth map (linked geographic brushing, for previous research on this topic, see: (Monmonier, 1989; Dykes, 1997; Haug et al., 1997). For this initial prototype, we implemented a one-way communication from the scatterplot to the map in which user interactions within the scatterplot are propagated to the map. Specifically, users can define class breaks on the x-axis of the scatterplot (the one depicted in the linked map) by dragging a vertical line that represents each break point. These breaks are then immediately propagated to the map display, with the color assigned to each state altered to reflect its new class assignment. Thus users can define classes based on the characteristics of one variable and evaluate the impact of those class break choices on the spatial pattern of that variable. [Go here to test drive the interactive version.](#)

Our initial progress in the Flash 4.0 environment demonstrated the feasibility of developing linked displays that include statistical maps. One drawback that we encountered was the inflexibility of loading new data into the interface through Flash 4.0's limited support for robust data structures. The highly customized nature of the individual components (scatterplot, choropleth map) also constrained the future application of these forms to new interfaces.

3.1.2 User-controlled animation sequences

A second prototype interface we designed combines the linked brushing concept with user-controlled animation sequences and more sophisticated data access. The interface is designed to allow users to load and explore county-based thematic data attributes through a cartographic display. Variables are loaded using eXtensible Markup Language (XML) syntax (a new feature supported in Flash 5.0). Since XML is rapidly becoming a standard information exchange language, supported by commercial databases, support for XML makes it possible for Flash to access a database efficiently. In our case, an Oracle database server stores the relevant datasets (see section 3.2 for details on database access issues).

The dynamic environment illustrated in figure 2 is designed to interpret the structure of an incoming XML file and, subsequently, construct visual and data elements from this code dynamically. In this prototype, once the data are processed by the Flash Player, they are stored locally and may be explored and visualized without further communication with the server.

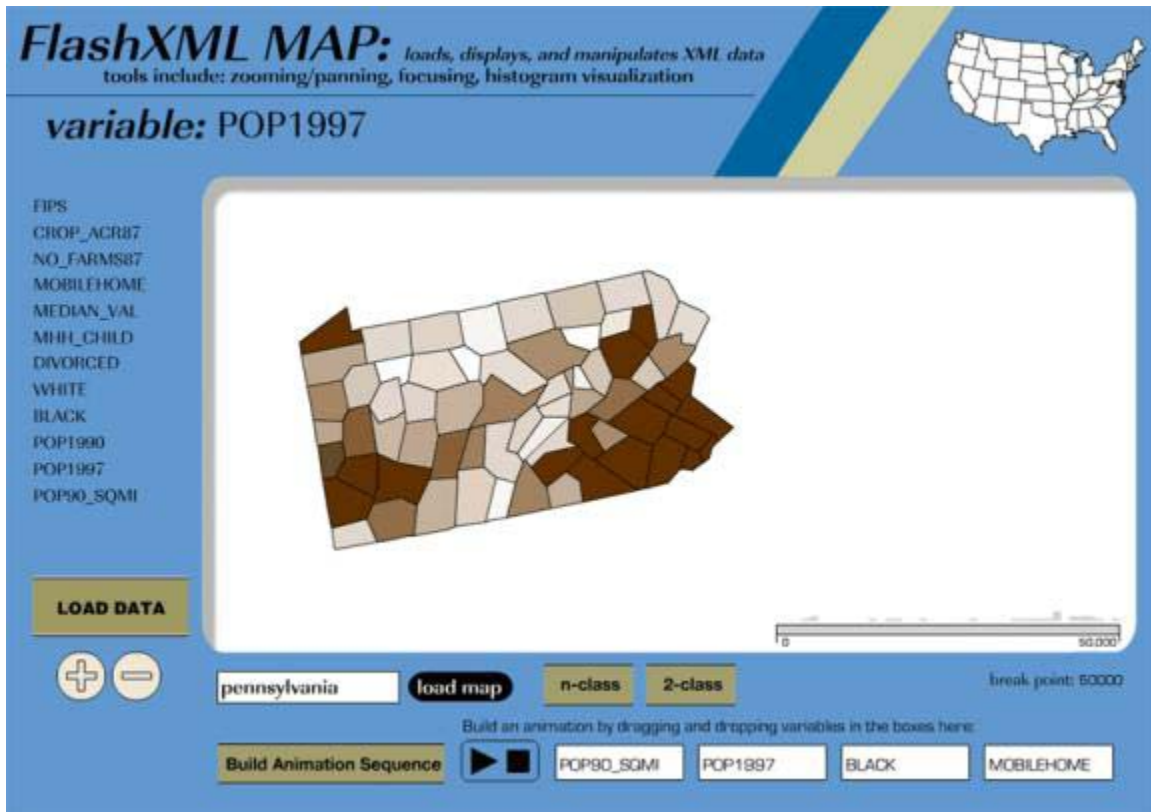


Figure 2. Prototype 2: XML-based map incorporating animation sequences

In prototype 2, three visual representation forms are generated upon data input: a choropleth map, a histogram display, and roll-over buttons for each variable. The number and names of the variables are defined based on the XML data. The prototype includes standard zooming and panning functions (for the map) and the three elements of the interface are dynamically linked. For example, if a user clicks or 'mouses over' a variable button, the attribute selection is immediately propagated to the map and histogram and if the user interacts with the histogram, the changes are immediately propagated to the map. Users can visualize each individual variable alone or may choose to rapidly shift between variables to compare attribute values and distributions.

The interface lists each geographic variable defined in the XML file in a column to the left of the map display. Upon arrival, the data associated with each variable are distributed to the geographic entities (e.g., counties), who control their own display properties. When the user chooses a new variable, each geographic unit now 'knows' to refer to the chosen locally-stored attribute. Thus, data are accessed simultaneously by each object without referring to the original dataset or making a new call to the server.

Similarly, the histogram object recognizes the shift to a new variable and will immediately update its form based on the user's selection.

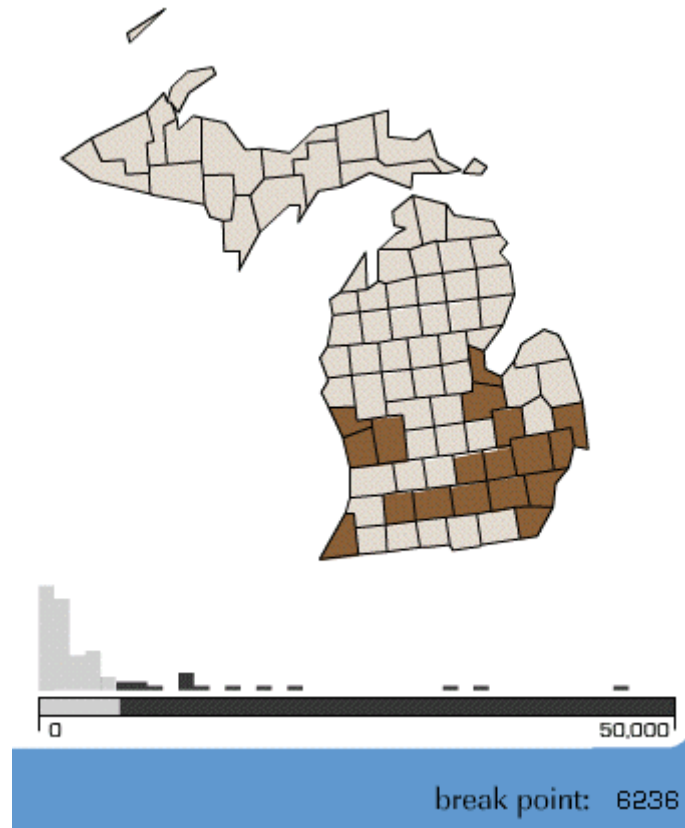


Figure 3. Prototype 2: Histogram-Map link with draggable class break

The histogram supports linked geographic brushing in the form of a moveable class break similar to the scatterplot described in the first prototype. The user may click and drag along the base of the histogram display to create a two-class representation of counties above and below a certain class break. Users can also define a break point for a single variable and then jump among the different variables to compare the spatial distributions of the variables based on this cutoff criterion. The dynamic interaction between a statistical distribution representation (histogram) and a cartographic representation (choropleth map) may allow users to explore geographic datasets in greater depth with the possibility of generating novel hypotheses.

A principal addition to the second prototype was the implementation of a tool to create user-defined animation sequences. This tool allows users to 'drag and drop' variables

from the variable column into steps of a four-part sequence. Users can thus build animated sequences of temporal or non-temporal attribute series. Simultaneously, they can interact with the histogram display to produce a dynamic brushing effect with a single class break that is transmitted to each variable in the animated sequence.

The second prototype demonstrates the feasibility of using dynamically-loaded XML files to create visual forms and interactive graphics. While the interface is extremely lightweight (114k), the data loading process was cumbersome and sometimes produced errors. Once the data were loaded, the system response was immediate and satisfactory, however, and we expect to be able to make progress in optimizing the loading process.

3.1.3 U.S. by county - interacting with detailed maps

In the first two prototypes, we dealt with both a limited number of variables and limited number of geographic entities. To further explore the capabilities and limitations of Flash as an environment for web-based geovisualization (particularly in support of the missions of our Digital Government partner agencies), we created a new interface for county-level data of the entire United States. Our most recent prototype was designed to evaluate Flash's ability to handle large datasets both in terms of data import and interactivity. We used the same process of creating geographic units as we did in prototype 2. Generating the base map to accept the XML data was the most labor-intensive process of the design. Fortunately, we can now use this generic component in part or in its entirety to apply to later interfaces.

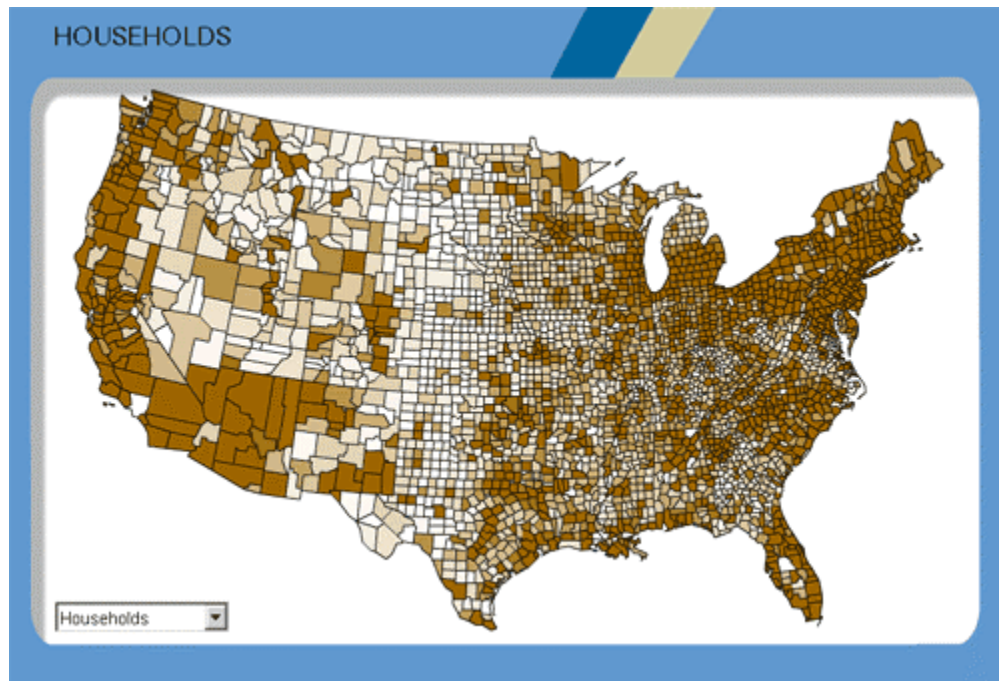


Figure 4. Prototype 3: U.S. by county, loaded through large XML file

While we streamlined our code to perform only absolutely necessary operations on the dataset, we saw significant slowdowns in system performance on the county-based interface. An XML file containing fips codes and a single attribute took about one minute to load and the system response once loaded was unsatisfactory for linked brushing and animation. This is a recent prototype and we are currently evaluating our interface in an effort to speed up the processing.

Flash provides support for interpreting and constructing XML files to allow efficient and robust client-server communication with small datasets. Flash's basic XML parser allows the designer to build interpreters for the expected XML file in order to store attribute names, values, and metadata. The following section details the database side of the prototype designs we have begun to implement as described above.

3.2 Database access

A Flash client is limited to processing a rather small set of data at one time. To support visual exploration of a wide range of data, we are building a database of demographic, health, and related statistics on a dedicated Oracle database server. To make the system functional for more than a trial set of embedded data, it is necessary to develop flexible strategies that allow web users to explore the large database with a lightweight Flash client.

There are several advantages to supporting a dynamic link between the Flash client and the database server. Among them: 1) The user can freely select and focus on any interesting subset of data for exploration; 2) with a centralized database, we eliminate many problems regarding consistency checks, maintenance, and updates of data; 3) through a standard interface (here we adopt the XML format as the communication syntax), the visualization modules and database management are separated, which can ease the evolution of both sides (e.g., allowing us to easily replace the Flash-based geovisualization modules with ones written in Java). Due to the limited ability for the Flash side to process the data (particularly if we are to achieve the desired light-weight client), we also need the server to do most of the data query and processing tasks and then serve the client with a small ready-to-visualize dataset. We evaluated two technical solutions to this dynamic connection. One is through Java Servlet technology. The other is to develop a dedicated server in Java language.

3.2.1 *Connection through Java Servlets*

A servlet is a Java class used to extend the capabilities of web servers. Servlets provide a component-based, server- and platform-independent method for building web-based applications, without the performance limitations of CGI programs. So, a servlet can be thought of as an 'applet' (a Java class that runs in web browsers) that runs on the server side—without a 'face'. It works as follows (figure 5).

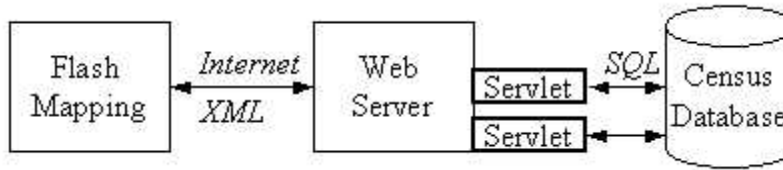


Figure 5: Connection through Java Servlet.

The Flash client can send a request like:

<http://rangoon.geog.psu.edu/servlet/OracleConn?tablename=counties&statername=Pennsylvania&attribute=pop1990,white,black>

Here, <http://rangoon.geog.psu.edu/> is the HTTP address of the web server, OracleConn is the name of the servlet, tablename=counties &statername= Pennsylvania &attribute= pop1990, white, black is the parameter string for this request. The OracleConn servlet can parse these parameters and compose an SQL query:

Select POP1990, WHITE, BLACK from COUNTIES where statername= 'Pennsylvania'.

The servlet will then connect to the database through JDBC (Java Database Connectivity) APIs, execute the above query, and get the data back. In our implementation, in addition to retrieving data, the servlet also preprocesses the data (e.g. calculates the mean value, median value, and other information) as required by the Flash client. Then the retrieved data and the derived information will be packed in XML format and passed to the client. From the Flash client's view, this request is exactly the same as downloading an XML file from the web server—it does not know or care how the file is generated. To be aware of what data are available in the database, the Flash client needs first to retrieve the metadata from the database before issuing any query requests. The metadata is also passed in XML format.

3.2.2 Connection through a dedicated Java Server

Another choice for implementing the connection is to build a dedicated server in the Java language. Adopting this strategy, we built a mini server that does not require embedding the service in a web server (as we did with the Java servlet). The communication protocol is still XML. With our own server, we have the flexibility and freedom to design a more complex protocol than the parameter string used by Java servlets. For this implementation, the Flash client will first build a socket connection with the server. Then all the communication (including query, metadata, and data) between the Flash client and the Data server will be XML messages sent back and forth through this socket.

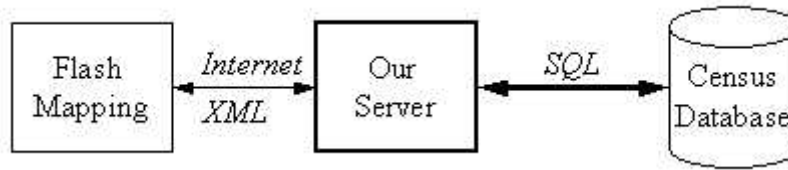


Figure 6: Connection through a dedicated server.

For example, our server runs on the machine `rangoon.geog.psu.edu` at port 2001. The Flash client first builds a socket connection to this server. Then it can issue a query such as:

```

<select>pop1997, white, black</select> <from>COUNTIES</from>
<where> <statename>Pennsylvania</statename> </where> <order> pop1997 </order>
<derive> mean, median </derive>
  
```

The communication protocol can be more complex than this; it is all up to the agreement between the client and the server. The server then parses the above request into a SQL statement:

```

Select pop1997, white, black from COUNTIES where statename = 'pennsylvania' ORDER BY pop1997
  
```

Then it will calculate the mean and median values of the data (as `<derive>` tag pair required) and pass it to the client in XML format.

3.2.3 Comparison of two techniques

There are both advantages and disadvantages for the two alternative forms of client-server connection described above. These advantages and disadvantages are outlined for each below.

With a Java servlet, it is easy to implement and deploy a new service. However, the request-response communication is limited to the parameter string. Thus, it is hard to express a complex request. The servlet methods seems suitable for usability experiments, each of which is not too complex and needs quick implementation. Another advantage of a Java servlet is that it is embedded in web servers and can be accessed easily through standard http protocol and the standard port 80, which is accessible through almost any firewall. As security issues become increasingly important in universities and government agencies, this advantage may become an overriding one.

With a stand-alone Java server, it is possible develop more flexible and complex services, while being no more difficult to build. One example would be a collaborative

environment in which distributed users could interact and observe interactions on the same dataset. Another drawback we have encountered is that many firewalls are configured to limit access to a small number of standard ports (ftp, http, etc.), which means the server used in the example above (port 2001) cannot be accessed outside of the local firewall.

4 DISCUSSION

The DG project reported here is focused on improving the quality of maps and graphics available in geovisualization tools for the web and to address the usability of these tools through implementing and assessing a series of interface prototypes. Our broad goals are to support sophisticated exploratory graphics for experts and comparable statistical summaries in a form that is available to non-expert public users through web-based interfaces. We have implemented three prototype interfaces in Flash to illustrate and assess the effectiveness of specific geovisualization concepts and to examine the extent to which Flash can support rapid prototyping and cognitive experimentation.

Developing flexible and robust geovisualization tools for the web is a challenge. Meeting this challenge often requires trade offs between functionality and ease of construction. Developing effective map-based interactive graphics 'from scratch' using a formal programming language such as Java or C++ requires considerable programming skill and is often frustrated (even for those with such skill) by the limitations of these languages when faced with implementing subtle variations needed to support good graphic design.

Designing in Flash holds some distinct advantages over existing development techniques in both prototyping and producing functional geovisualization interfaces, not the least of which is design time. Flash designers with limited programming knowledge can produce functional and graphically rich interfaces very quickly for prototyping and testing conceptual extensions. The Action Script interface is intuitive enough for inexperienced programmers, while flexible enough to support complex development by expert designers. These qualities make Flash well suited to graphics prototype development and may support fairly sophisticated products in the future. Flash also dramatically improves the graphical flexibility of the designer to apply existing knowledge we have of cartographic theory to interactive displays. Furthermore, Flash is lightweight and is run on the web through a small plug-in that has achieved very wide distribution (estimates suggest that 90% of current users have at least the 4.0 plug-in installed in their web browser). The transition from development environment to the web is as simple for designers as choosing 'Publish' from the Flash design interface. Finally, our examples illustrate the ability of Flash to support a database connection through a dedicated server or a Java servlet. Once a connection has been made to the database, the data may be stored locally and will support multi-perspective interactions without further database queries.

Although Flash has many features that make it attractive for web-based geovisualization applications, we have also encountered some serious limitations. The main drawback of designing geovisualization applications in Flash is the inability of the software to recognize any existing spatial data formats. The geographic base for the maps produced in these three prototypes were created manually—a process that allows us to prototype county- and state-based data, but is not efficient for examining new geographies. We also observed a considerable processing slow-down for loading and displaying large datasets (one minute vs. ten seconds). This limitation may become increasingly restrictive as we begin to evaluate our county-based US map (3000+ entities) while adding further interactive controls and animation.

Despite the disadvantages listed here, there is considerable interest for our group in exploring Flash for potential future applications and testing. We are currently running cognitive experiments on a fourth prototype designed to inform our future applications of animation within our geovisualizations. Designing these prototypes in a lightweight environment such as Flash has the potential to speed up development and improve the graphical integrity of exploratory geovisualization tools in general. The rapid prototypes allow us to explore geovisualization concepts enough to assess their feasibility of construction and to afford formal cognitive and usability experimentation before committing costly resources to their development in a robust Java environment.

Acknowledgement and disclaimer

This material is based upon work supported by the National Science Foundation under Grant No. 9983451. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. Thank you to Luis Mejias for his efforts in building the geographic database.

References:

Andrienko, G.L. and Andrienko, N.V., 1999. Interactive maps for visual data exploration. *International Journal of Geographic Information Science*, 13(4): 355-374.

Dykes, J.A., 1997. Exploring spatial data representation with dynamic graphics. *Computers & Geosciences*, 23(4): 345-370.

Harrower, M., MacEachren, A.M. and Griffin, A., 2000. Design, implementation, and assessment of geographic visualization tools to support earth science education. *Cartography & Geographic Information Systems*, 27(4): 279-293.

Haug, D., MacEachren, A.M., Boscoe, F., Brown, D., Marrara, M., Polsky, C. and Beedasy, J., 1997. Implementing exploratory spatial data analysis methods for multivariate health statistics, *Proceedings of GIS/LIS '97. AAG*, Cincinnati, OH, Oct. 28-30, 1997, pp. 205-212.

MacEachren, A.M., 1998. Cartography, GIS and the world wide web. *Progress in Human Geography*, 22(4): 575-585.

MacEachren, A.M., Hardisty, F., Gahegan, M., Wheeler, M., Dai, X., Guo, D. and Takatsuka, M., 2001. Supporting visual integration and analysis of geospatially-referenced statistics through web-deployable, cross-platform tools, Proceeding, dg.o.2001, National Conference for Digital Government Research, Los Angeles, CA, May 21-23, pp. 17-24.

Monmonier, M., 1989. Geographic brushing: Enhancing exploratory analysis of the scatterplot matrix. *Geographical Analysis*, 21(1): 81-84.
