Learnability of Interactive Coordinated-View Visualizations

Sujatha Krishnamoorthy, Christopher North
Department of Computer Science, Virginia Tech, Blacksburg, VA 24060 USA
{sukrish2@vt.edu, north@cs.vt.edu}

Abstract

This paper examines the Human Computer Interaction issue of learnability of interactive coordinated-view visualizations. We take the case of DataMaps, a Census data visualization tool intended for a general audience with a huge percentage of novices. Usability tests conducted on DataMaps revealed three main kinds of problems that novices faced: they could not make strategic selections of coordinated visualizations according to a given task, they lacked familiarity with the nature of the attributes, and there were several misunderstandings of visual syntax and interaction widget usage. We outline design features which are desirable for novice-friendliness: Task based organization of coordinated views to enable strategic selection of views to suit the task, Data centric approach to familiarize novices with data, Self disclosure of visual syntax features and interaction mechanisms by the interface. The design should be such that they can smoothly transition from being a novice to expert. We examine how these principles may be applied to DataMaps to re-design it for "novice-friendliness".

1. Introduction

The learnability of interactive, coordinated-view visualizations is an important issue in several scenarios. Consider a visualization interface in a museum or a website, available to a varied audience. Several users might be accessing the visualization tool only for a one-time exploratory session. A visitor in a museum might want to interact with the display for a while. Novice-friendliness is crucial in such cases and we are justified in researching learnability issues in visualizations.

We define a novice user as someone who does not have experience with using interactive multiple-view visualizations. However, we do need to assume a "lowest common denominator profile" for novice users to design the interface with respect to that profile. We assume that a user, although novice, necessarily possesses:

- General familiarity with the notion of using visual representations of information (as opposed to textual paragraphs). For example, they would be familiar with visual representations encountered in day-to-day life, such as pie charts and bar graphs and weather information on maps as seen in weather news broadcasts.
- General familiarity with W-I-M-P interfaces (Windows Icons Menus Pointer interface) where the user must rely on icons, buttons, and dialog boxes for executing operations.

We focused our studies on a particular visualization tool called DataMaps, supported by US Census Bureau. We expect that a typical user browsing the Census website would be familiar with WIMP interfaces. An informal survey revealed that bar-charts and pie-charts are the most commonly encountered and easily understood visualizations for those who have no prior exposure to interactive information visualizations. It also showed us that it is acceptable to assume basic familiarity with the notion of using visual representations in novices.

In sections 2, 3 and 4 of this document, we provide a description of DataMaps, the usability test conducted on it and the qualitative results obtained, respectively. In sections 5 we theorize about learnability issues and in section 6, we outline design principles which enhance learnability. In section 7, we show how the design principles may be applied to re-design DataMaps for novice friendliness.

2. Background on DataMaps

DataMaps [2] is a front-end tool for visualization and analysis of census data on the United States. Data has been collected for approximately 8000 attributes. "Attributes" are items such as "total population" "percentage white population" "percentage black population" and so on. These attributes are grouped together by categories such as "Age", "Agriculture", "Banking"," Crime" etc. See figure 1.

DataMaps has a map view, a histogram view, a table, and a scatter-plot view. The histogram shows a frequency distribution (the number of regions occurring in predetermined class intervals). Several histograms are simultaneously visible. The map view employs color gradation to show attribute values as a function of geographic areas. Darker regions imply lower numerical values. Only one map is viewable at a time. The scatter plots aids in assessing the relationship between two attributes. Only one scatter-plot is present. The table is activated on clicking on a region on the map; it loads data values for that state and presents the textual figures in tabulated format. The coordination mechanisms are described below:

- The histograms are present together with a Dynamic Query (DQ) slider widget. Moving the sliders in the dynamic query widget filters out value ranges, and the corresponding states in the map are de-selected (darkened out), as shown in figure 1. (de-selection on DQ slider -> de-selection on map, unidirectional)
- Clicking on a region on the map loads the table with figures for that state. Refer figure 1.Georgia and Florida are selected in map and tabulated in table view. (selection on the map -> loading in the table, unidirectional)

- Clicking on the map also highlights the corresponding dot on the scatter plot, and clicking on the scatter plot highlights the region on the map. The two dots for Georgia and Florida are seen highlighted on the plot, in figure 1. (selection on map <-> selection on scatter plot, bidirectional)

When the application is opened, five attributes are selected by default. Five corresponding histograms show the frequency distributions for these attributes. If the user needs to visualize attributes other than these five, he/she has to click the “more variables” button . A tree-view of the 8000 attributes (as seen in Figure 1) would appear on a separate window and allow the user to select new attributes to view.

3. Usability tests on DataMaps

The Census department conducted usability studies on DataMaps. Novice users were given 10 tasks and asked to think aloud while trying to complete each task. They were asked to “play around with the interface” as a pre-task session and describe their first impressions. The sessions were video-taped and screen-captured. All of users were quick to remark that they would click on a state to view information about it, within only a few seconds of playing with the Data-maps interface. Table 1 lists the actual tasks that the novice users were required to do.

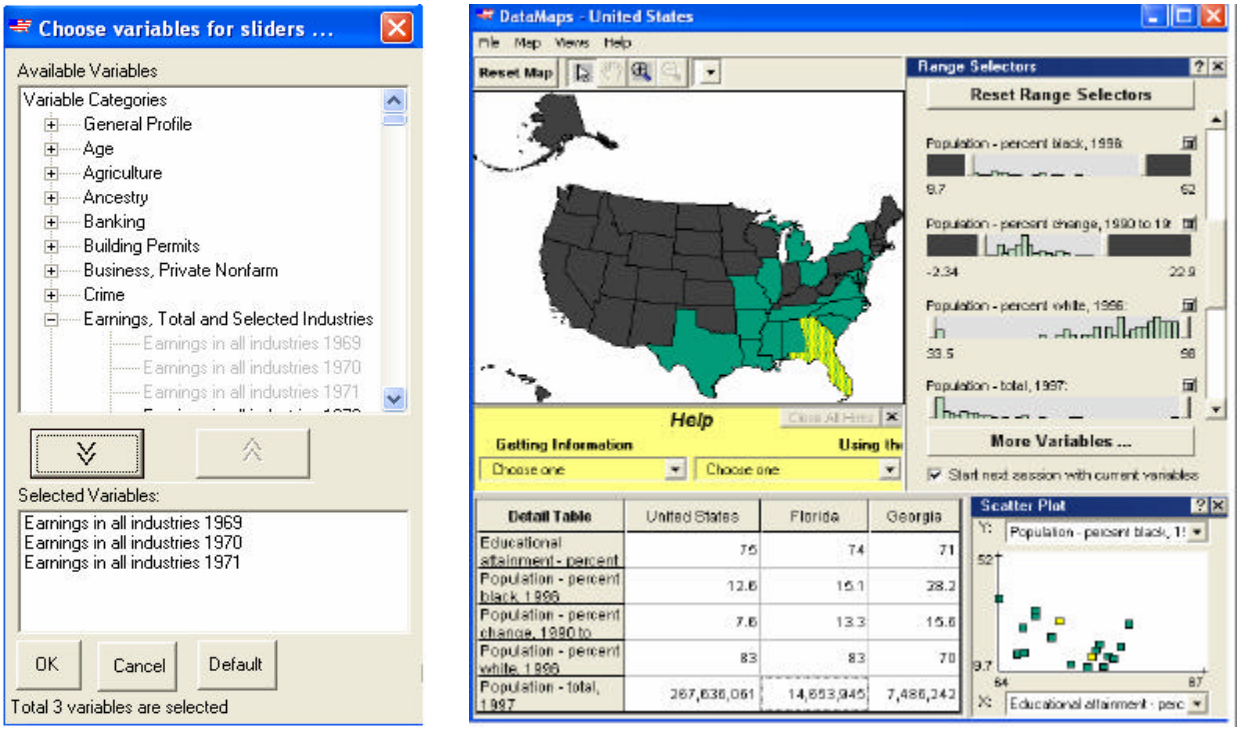

Figure 1 Tree View and Data Map windows

S.N	Question
1	What was the population of California in 1997?
2	Compare the 1994 income level of people in 2 states not next to each other. Which had greater per capita income?
3	Of the western states, which state had the lowest unemployment rate in 1996?
4	Name the states where 20% to 22% of the population in 1990 had high school degrees.
5	Which two counties in Nevada had the highest population percent change from 1990 to 1997?
6	How many families in Minnesota lived below the poverty line in 1989?
7	You are thinking of moving to a new state. You want to live in a state that has low poverty level, high income level, and low unemployment rate. Which state or states best fit these criteria?
8	You are interested in graphing the relation between high school graduation and one's personal income across the states. How would you do it?
9 a	Which county in the USA had the highest population in 1997?
9 b	You would like the map to show number of persons below poverty level for each county. How would you do that?
10	You no longer want to play around with one data item. How will you remove it from your view?

Table 1 Task set in DataMaps usability test

4. Usability problems identified

4.1 Inefficient strategies

When faced with a question such as "which county in the USA had the highest population in 1997", they tried to click each state on the map and read off corresponding values for each region from the table. In order to locate the maximum they tried to follow a laborious, algorithmic procedure. It required excessive scrolling of the table and mental book-keeping; and they gave up. Simply selecting the given attribute to color the map would have helped locate the maximum valued region by color tone, but they did not follow this method.

4.2 Understanding interaction widget mechanisms

The users had not seen Dynamic Query sliders before. The sliders resemble buttons; they have been marked with arrows in figure 2. The users clicked on each one, but they couldn't guess that they must be dragged in order to be used, not just clicked. Dragging selects a sub range and filters out (deselects) states not in that sub-range. This operates similar to Dynamic Queries in Home Finder visualization [6].

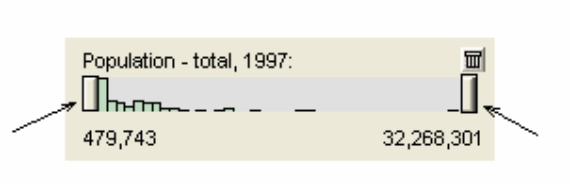

Figure 2 Dynamic Query widget

4.3 Understanding visual syntax

The users thought the histograms were bar graphs. They had trouble interpreting the visual syntax. In a histogram, the height of the bar stands for number of occurrences, but they wrongly interpreted that a bar stands for a particular state and that the height of the bar represents the attribute value for that state. The y-axis of the histogram was left unlabelled to save space and accommodate more histograms as seen in figure 2. If labeling were clear, this problem would have been eliminated.

4.4 Locating starting points of action sequences

Two of the users had trouble finding out how to visualize information for new attributes, not already visible in the interface. Locating the "more variables" button which opens the tree view of attributes was a problem (see figure 1). This would have been eliminated if the tree view had been present persistently on a separate frame.

4.5 Difficulty breaking-up the question into executable tasks

Consider question 6 from the table: "How many families in Minnesota lived below the poverty line in 1989?" One could break down the question in several ways. For example:

- "How many families" + "lived in Minnesota" + "below the poverty line" + "in 1989"
- "How many families" + "lived below poverty line in 1989" + "in Minnesota"

The attribute that actually bears the answer to the question, is: "Number of families below poverty line, 1989". The value for this attribute should be checked against the state "Minnesota". Of the two types of break-downs, if the latter were to be used, then the user has a better chance of finding the required attribute, and looking up Minnesota against it.

We need to make sure that the user's mental model matches the system model. One user tried to tackle the question by first figuring out the value of "the poverty line" in dollar amount and then counting the regions with values below that. There was no piece of information about "the poverty line" and his strategy failed. If the attribute list were persistently present, it would have been easy to see the required attribute: "Number of

families living below poverty line in 1989". He would have better understood how to break down the given task.

5. Learnability issues

We break down the learnability issues that we identified into the following broad categories:

- **Strategy** - Breaking down higher level goals into an action plan with the best strategy.
- **Data familiarity** - Understanding the nature of data attributes, data values, and meta-data information.
- **Representation and Interaction** - Understanding the visual syntax and interaction mechanisms.

At first, we theorized that the novice users used the inefficient strategy of relying on tabulated numeric values for all problems, probably because they are more familiar with tables and uncomfortable upon seeing other visualizations. However, related literature led us to consider another theory. Fu et al examine novice users' learning behaviors in "Probing the Paradox of the Active user: Asymmetrical Transfer May Produce Stable, Suboptimal Performance" [4] and "Resolving the paradox of the active user: stable suboptimal performance in interactive tasks" [5]. They explain why users persistently use inefficient methods for completing tasks when the users know more efficient methods.

It may be that the users are simply resorting to the first strategy they learned, to solve all problems. In the test, the first question obviously leads the user to play with maps. The users automatically became familiarized with clicking regions on the map and viewing the corresponding details on the table. "Click California, read off value from table" was the strategy followed.

Now the map has four different kinds of functionality in DataMaps:

- To act as a pointer to a table row (clicking on a state loads the table with figures for that state)
- To act as a visualization of a single attribute (color gradation shows attribute values as a function of geographic areas)
- To act as an output to dynamic queries from histograms (as described earlier)
- To act as a two-way selection with scatter-plot. (described earlier)

Of these four roles that the map plays, the first task made the user recognize the map as a pointer to a table. So the first "strategy" that they learnt was: "Click on a region in the map, and view corresponding values on the table". They "latched on" to the learnt strategy. So the users kept on trying to answer all questions using the same strategy. When they came to questions such as (9a) which asked them to identify the region with maximum value for a given attribute, they were frustrated, thinking that they had to click on 50 states, one by one, and read off values from the table to identify the state with the maximum value.

Fu et al refer to the user's inefficient but most often used method for completing a task as a user's "preferred method" and the more efficient but less often used method for completing a task as the "recommended method." They explain that users may find incremental actions to be less of a cognitive load. Interaction with modern user interfaces is usually done through incremental steps. From their explanation, it might appear that users are doing what brings them the least amount of stress and smallest cognitive load; repeating simple tasks many times. This offered an explanation as to why users tried to use the same strategy of "click on state, scroll through values on table" to solve all the questions.

The users had difficulty breaking up the tasks and tackling them because they weren't seeing the data first. They were not familiar with the nature of data attributes before needing to use them. They were seeing the questions first. The data attributes occur on yearly basis. For example, "total population, 1997" "total population, 1998" etc. Such observations about the nature of the data can be understood only by being continually exposed to the tree view listing of data attributes.

People are most familiar with the notion of data and bring with them their prior knowledge, as novices, to the visualization tool. Leveraging their existing knowledge to promote learnability is possible by making the interface data centric. Data attributes and their relationships with each other and meta-data information can be made prominent.

Upon mouse-move over the slider buttons, the mouse pointer could have changed shape to indicate "drag-ability". This would have self-disclosed the mechanism by which it was meant to be used, without the need of help files. The y-axis on the histogram could have been labeled prominently. Users may not be familiar with histograms, although they may be familiar with bar charts. Currently, the knowledge about how to use widgets is imparted via help mechanisms, rather than being embedded during task execution.

6. Design principles for learnability in DataMaps

6.1 Data Centricity

"What can I visualize with a map?" – is visualization centric (the old design of DataMaps). Data driven action sequences would "make sense" to the user.

"What is the data they have collected, and how can I use them to get my queries answered?" is more natural, and is the new design. As described in section 4.5, prominently displaying the attribute list helps make the mental model closer to the system. The process of breaking-down a task into steps to perform the query is easier when the attribute names are clearly visible. Meta-data information should also be available upon demand.

6.2 Task based configurations of multiple views

Novices need to learn how to persistently use efficient methods for performing tasks. According to T. Bosser [1] these must be taught to the user initially, if they are to be used persistently. Novices tend to stick to initially learnt strategies. So overloading the same visualization component with too many task capabilities (e.g. the four roles of the map described earlier) might result in the visualization being used only in one of the roles which was initially learned. According to task-based principle, a map component shall show color gradation only when the task objective is to geographically visualize attributes using a color gradation. A map shall remain un-shaded while acting as the output of a dynamic query made from sliders; it will merely show de-selections. These two different roles are kept separate in different configurations. Only one configuration can be kept open at a time, so the components will be used appropriately for the task at hand. The system should explicitly describe the purpose of each task-based configuration.

Task based organization may be criticized that it limits what the user can do or that it "over-trains" novices in certain task types and they may fail to use visualizations to creatively discover insights. To offset this drawback, the snap together [7] principle may be accommodated as well. This is explained in section 7.

6.3 Self-disclosure

Help menus are a common but ineffective solution. Users seldom read through them before starting to use an interface. John Reiman finds that many users feel that exploration or exploratory learning is an ideal way to learn how to do tasks [8]. Exploratory learning is a method of learning that allows users to figure out how to perform tasks as they are doing them. DiGiano and Eisenberg [3] describe self-disclosure as the process of embedding context sensitive help into learning opportunities. DiGiano and Eisenberg provide an excellent example in [3] by describing a feature in AutoCAD that bridges the gap between mouse driven commands and text programming commands. It worked like this: When a user performs a function with a mouse a window in the AutoCAD program displayed the text programming command equivalent of the mouse command. We shall define self disclosing interactive visualization as one which discloses elements of its visual syntax and interaction mechanisms of its widgets, during actual task execution, without requiring the user to consult external manuals to comprehend it.

7. Re-design of DataMaps for learnability using the above design principles

The basic re-design is discussed, without too many implementation details. The DataMaps interface can be divided into two major regions – data attribute explorer region and the visualization region. The attribute list shall be kept persistently visible on the tree view as shown in figure 3.

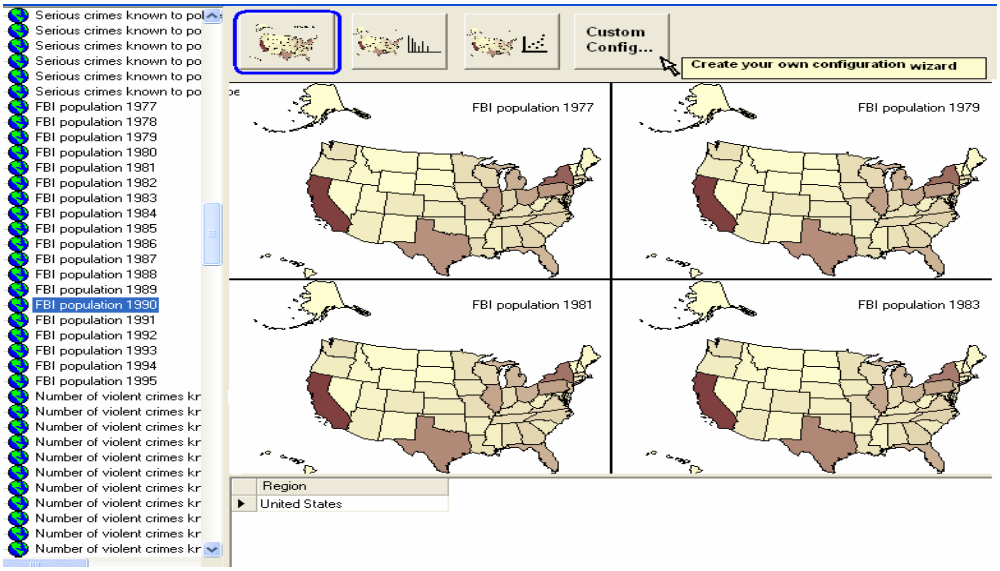

Figure 3 DataMaps redesigned for learnability

A panel on top of the DataMaps application has a set of buttons with icons on them as shown. Each button represents a configuration or arrangement of coordinated views that can be used to answer specific kinds of queries. For example, we can have "maps only configuration" to view one or more color scale maps, "map and Dynamic query widget" configuration to view regions satisfying input query conditions, "map plus scatterplot" configuration and so on. On moving the mouse over the buttons, tool tips appear, describing the configuration.

When DataMaps application is opened, a particular configuration of visualization tools is shown, by default as shown. A note explains what the configuration may be used for. For example, the note in the "maps only" configuration could say: "You can view colored maps for up to 9 maps at a time. This feature is useful especially for geographically comparing several different attributes". This note may be a separate window or a frame in the overall DataMaps window.

If the chosen configuration is "maps only", then every time an attribute is selected to be visualized, a new map is loaded into the visualization area. Each map would act as a color-scale visualization of an attribute. If the configuration chosen is "maps + DQ histograms" then the map would only serve as an output of DQ results. For example, "show states with population ranging from x1 to x2 AND crime rate ranging from c1 to c2" would be executed with "maps + histogram" configuration. The map would highlight states which satisfy the query input from the dynamic query widget. Only one configuration can be opened at a time. The button of the current configuration is highlighted.

The last button on the panel is "custom configuration…" It would take the user through a wizard of steps. The user would be asked to choose the visualizations and the type of coordinations between them, similar to snap together visualization [7]. After going through steps to construct configurations, the user will know what he/she is doing, and why.

The individual visualization components are clearly marked and labeled. Mouse-move-over events are carefully traced out so that all necessary prompting information is provided. For example, upon mouse-move-over on the Dynamic Query slider widget, mouse pointer changes to indicates that it is drag-able. The interactive self-disclosure of visual syntax and interaction mechanisms will help minimize the need for help documents.

The novice user is supported from the novice stage to the expert stage. The user is familiarized with the notion of "specific configurations for specific tasks". A complete novice can simply explore all the available configurations and read the corresponding explanatory notes. As the novice becomes more comfortable, he/she can go through the custom configuration wizard.

Conclusions and future work

We have used the results of empirical studies to generate design guidelines for enhancing learnability of an interactive visualization. Our next step is to implement the new design and run studies to collect further empirical data. We also wish to examine the applicability of the three design principles (Data centricity, Task based organization, and Self disclosure) to enhance the learnability of other visualizations of relational data.

Acknowledgements

We would like to acknowledge the work of Umur Yilmaz (graduated student, Dept of Computer Science, Virginia Tech) in conducting experiments with a persistently visible attribute tree view design. Positive usability results from his work provided reason for us to look at data centricity.

References

[1] T Bosser. Learning in man-computer interaction. Technical report. Springer-Verlag New York, Inc. (1987).

[2] DataMaps for census visualization. http://infovis.cs.vt.edu/census/Datamaps.htm Virginia Tech Information Visualization Research group. 2004.

[3] DiGiano, Chris, Eisenberg, Mike. Self-disclosing design tools: a gentle introduction to end-user programming. Conference proceedings on Designing interactive systems. p.189-197. (1995).

[4] Fu, W. Veksler, V.D. , Gray, W. D. Probing the paradox of the active user : asymmetrical transfer may produce stable sub-optimal performance (2004).

[5] Fu, W. , Gray, W. D. Resolving the paradox of the active user: Stable suboptimal performance in interactive tasks. Cognitive Science, 28(6). (2004).

[6] Jain, V. & Shneiderman, B. (1994). Data Structures for Dynamic Queries: An Analytical and Experimental Evaluation. Proceedings of the Workshop on Advanced Visual Interfaces. NY: ACM, 1-11.

[7] C. North and B. Shneiderman. Snap-together visualization: A user interface for coordinating visualizations of a relational database. Proceedings of the 5th International Working Conference on Advanced Visual Interfaces (AVI 2000), Palermo, Italy, May 2000.

[8] Rieman, John. A field study of exploratory learning strategies. ACM Transactions on Computer-Human Interaction (TOCHI). v.3 n.3. p.189-218. (1996).